AD-A110 898 LOGISTICS MANAGEMENT INST WASHINGTON DC F/G 15/5
THE SORTIE-GENERATION MODEL SYSTEM. VOLUME II. SORTIE-GENERATIO--ETC(U)
SEP 81 J B ABELL, R S GREENBERG MDA903-80-C-0554
UNCLASSIFIED LMI-DP101-VOL-2 NL

1 OF 2
AD
A110898

12

THE SORTIE-GENERATION MODEL SYSTEM
VOLUME II
SORTIE-GENERATION MODEL USER'S GUIDE

September 1981

John B. Abell
Robert S. Greenberg
Michael J. Konvalinka

Prepared pursuant to Department of Defense Contract MDA903-80-C-0554 (Task DP101). Views or conclusions contained in this document should not be interpreted as representing the official opinion or policy of the Department of Defense. Except for use for Government purposes, permission to quote from or reproduce portions of this document must be obtained from the Logistics Management Institute.

LOGISTICS MANAGEMENT INSTITUTE
4701 SANGAMORE ROAD
WASHINGTON, D.C. 20016

PREFACE

This volume is the second of six volumes that describe the LMI Sortie-Generation Model System. Volume I, Executive Summary, discusses the problem the system is designed to address and provides an overview of the principal parts of the system. Volume II, Sortie-Generation Model User's Guide, provides sufficient information to allow a user to run the Sortie-Generation Model (SGM). Volume III, Sortie-Generation Model Analyst's Manual, describes the mathematical structures, derivations, assumptions, limitations, and data sources of the SGM at a very detailed level. Volume IV, Sortie-Generation Model Programmer's Manual, specifies the details of the computer programs, file structures, job control language, and operating environment of the SGM. Volume V describes the maintenance subsystem and explains the construction of the maintenance input file to the SGM. Volume VI describes the spares subsystem and shows a user how to build the spares file that is used by the SGM.

Potential users are cautioned that no volume is intended to provide, by itself, all of the information needed for a comprehensive understanding of the operation of the SGM.

ACKNOWLEDGMENTS

We are indebted to Professor Edward J. Ignall of Columbia University for his many imaginative contributions to the formulation of the Sortie-Generation Model; to Professor Louis W. Miller of the Wharton School, University of Pennsylvania, who first suggested the basic structure of the multiple-server queueing system incorporated in the model; to Professor John A. Muckstadt of Cornell University for his helpful suggestions early in the model's development; and to our colleague at LMI, Mr. David L. Goodwin, who contributed greatly to the software development.

TABLE OF CONTENTS

LIST OF FIGURES

VOLUME II

USER'S GUIDE

1. MODEL DESCRIPTION

INTRODUCTION

The Sortie-Generation Model (SGM) is a hybrid analytic/simulation model that estimates the expected maximal number of sorties that can be flown by a specified aircraft type in a wartime scenario. This estimate is based on aircraft characteristics, maintenance manpower and recoverable spares levels, and user inputs that describe the scenario of interest. The purpose of this chapter is to describe the basic structure of the SGM. The reader who is interested in the technical aspects of the SGM is referred to Volume III.

THE STRUCTURE OF THE MODEL: STATES AND PROCESSES

The SGM consists of a collection of aircraft states, processes that cause transitions between states, and logic that governs those processes. The SGM simulates the transition of aircraft between these states throughout a daily flying schedule that is specified by the user. The definitions of the states, the logic of the state transitions, and the interaction of these transitions with the flying schedule determine the basic structure of the SGM.

Aircraft States

There are five aircraft states in the SGM:

1) Mission-capable
2) Maintenance
3) Not mission-capable, supply (NMCS)
4) Combat loss
5) Reserve

These states are mutually exclusive and collectively exhaustive; i.e., every aircraft is in one and only one state. The states are described below.

Mission Capable. An aircraft is considered mission-capable if it is capable of flying a combat mission. It is not mission-capable if it is undergoing essential corrective maintenance, or is missing a mission-essential part. There is no explicit representation in the SGM of aircraft that are partially mission-capable.

Maintenance. An aircraft is in maintenance status if it requires unscheduled, on-aircraft repair that is essential to the performance of its mission. This repair may or may not be due to failure of a part; however, in this model, an aircraft is not allowed to enter maintenance until all needed parts have been obtained from supply or repair.

NMCS. An aircraft is not mission-capable, supply if the aircraft is missing an essential part. In the SGM, only mission-essential Line Replaceable Units (LRUs) can cause an aircraft to become NMCS.

Combat Loss. An aircraft is counted as a combat loss if it does not return from a sortie. Once an aircraft has been lost it can never be recovered. Battle-damaged aircraft that return from a sortie are not considered in this model.

Reserve. Reserve aircraft consist of mission-capable aircraft that are used to replace combat losses. The user specifies an initial number of aircraft that are held in reserve at the beginning of the scenario; these reserve aircraft replace combat losses at the end of each day, until all reserves have been exhausted. Aircraft are allowed to leave this reserve state, but no aircraft can enter it.

Processes - Transitions Between States

There are eight processes in the SGM which cause transitions between aircraft states:

1) Ground aborts

2) Breaks

3) Aircraft repairs

4) Parts demands

5) Parts repair

6) Cannibalization

7) Attrition

8) Commitment of reserves

Figure 1-1 depicts the relationships among the various states and processes.

THE LOGIC OF THE MODEL: EVENTS AND THEIR SEQUENCING

The events that occur in the SGM are related to a flying schedule with user-specified characteristics. The flying schedule consists of a number of periods or cycles each of which is divided into three segments of lengths T_L, T_F, and T_W, respectively. During the last period, the T_W segment is replaced by an overnight recovery period. The user specifies the first and last take-off times of the day; the time, T_L, which is the average minimal length of time required to launch a sortie given a mission-capable aircraft; the sortie length, T_F; and the number of periods per day. The time, T_W, is then computed by the SGM program. The flying schedule is the same each day except for the number of aircraft to be flown each period, which the user can vary. A typical flying schedule is portrayed in Figure 1-2.

Figure 1-3 portrays two segments of a flying day; the flying period on the left is intended to be typical and the one on the right to be the last period of the day. The events that occur in the SGM are denoted by circled numbers placed under the figures at the appropriate positions on the time line. Each of those events is described here.

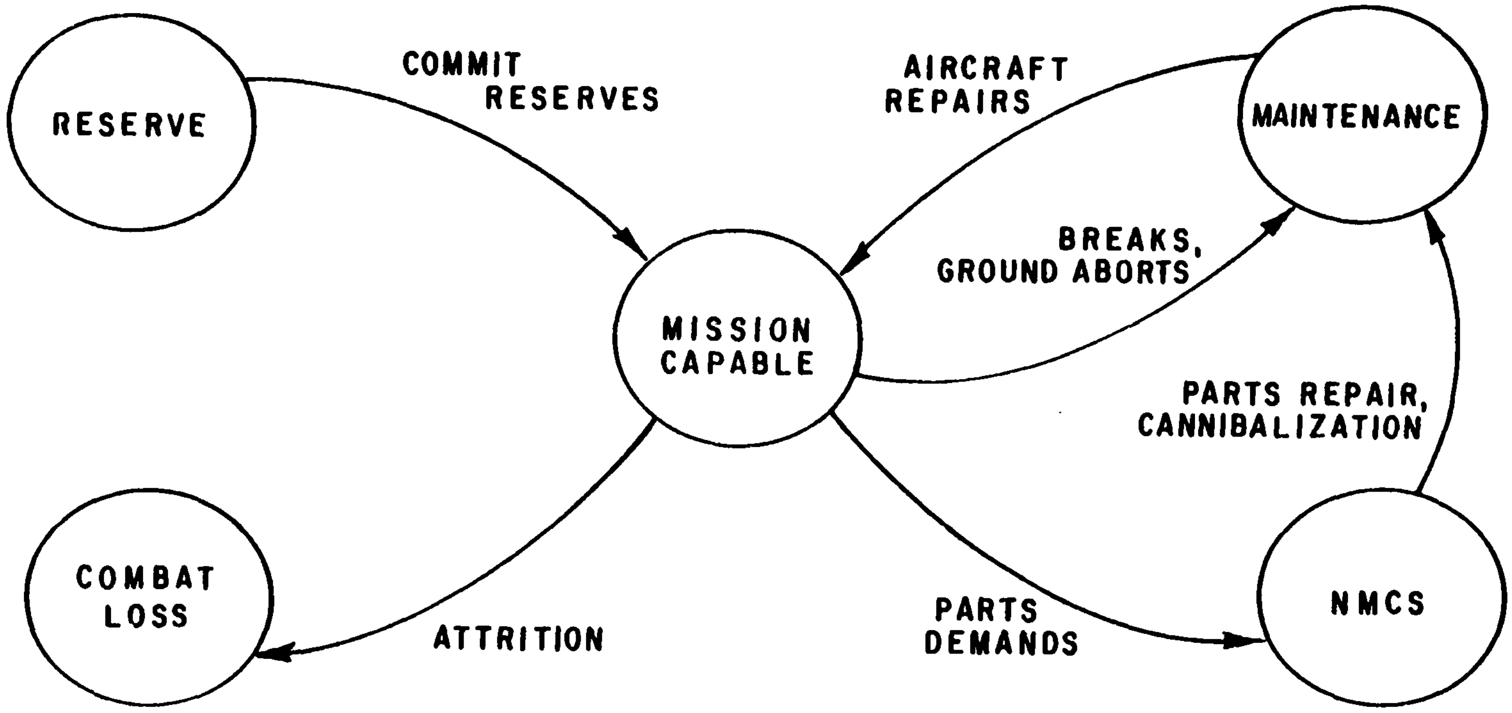
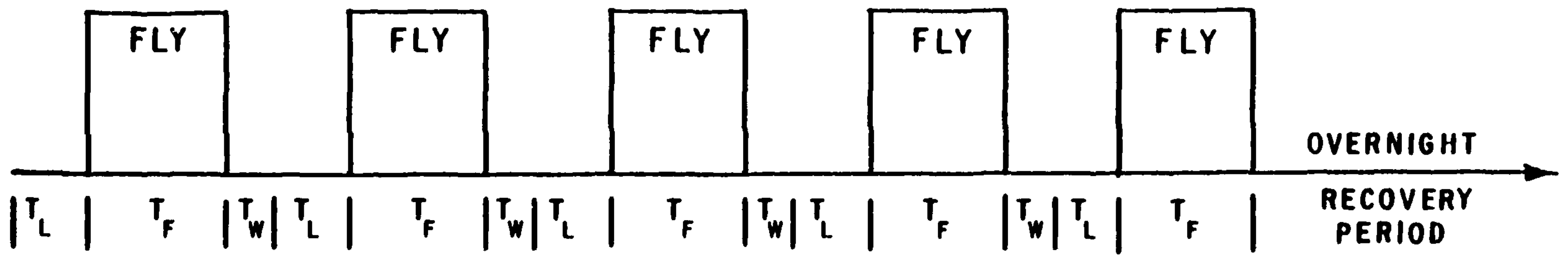


FIGURE 1-1
STATES AND PROCESSES



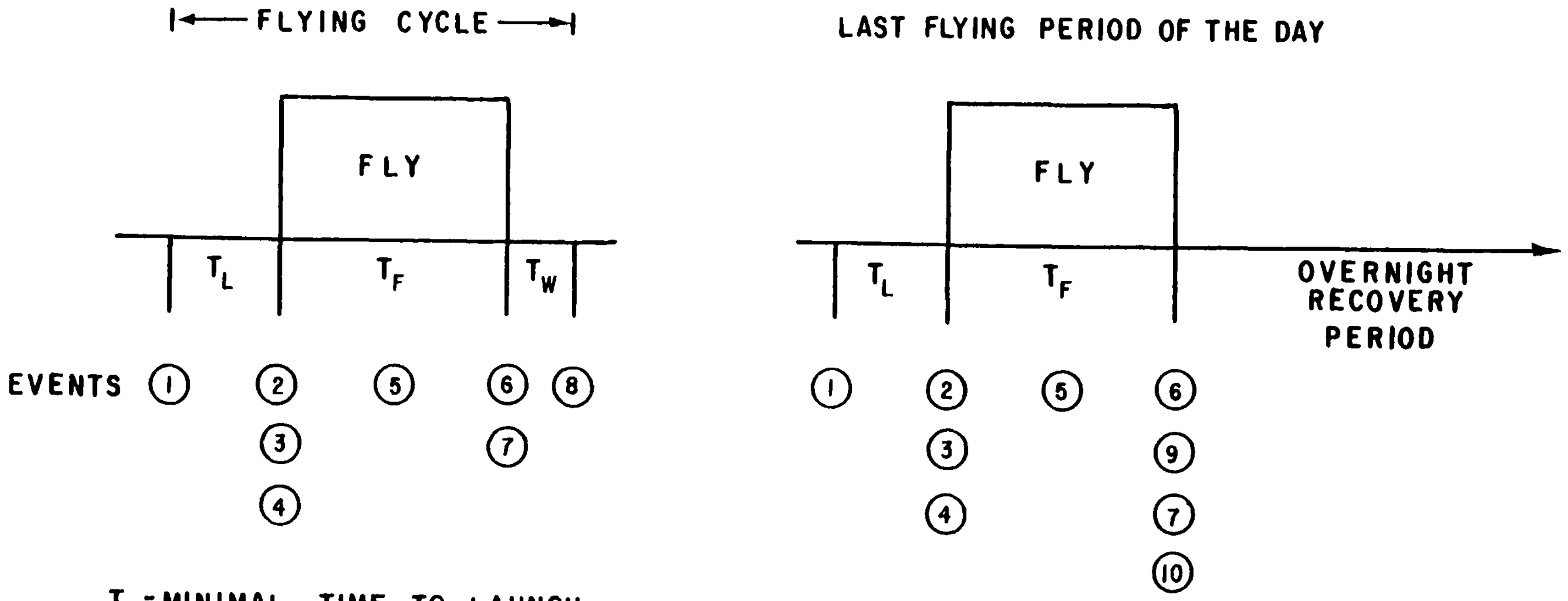
T_L = MINIMAL TIME TO LAUNCH GIVEN A MISSION-CAPABLE AIRCRAFT

T_F = SORTIE LENGTH

T_W = WAITING TIME (FIXED BY NUMBER OF PERIODS. FIRST AND LAST TAKEOFF TIMES, T_L AND T_F)

FIGURE 1-2

A TYPICAL FIVE-PERIOD FLYING DAY



T_L = MINIMAL TIME TO LAUNCH GIVEN A MISSION-CAPABLE AIRCRAFT

T_F = SORTIE LENGTH

T_W = WAITING TIME (FIXED BY NUMBER OF PERIODS, FIRST AND LAST TAKEOFF TIMES, T_L AND T_F)

FIGURE 1-3

SGM FLYING CYCLE

Event 1

All mission-capable aircraft are prepared for launch. Any aircraft that is not mission-capable at this time (i.e., T_L before takeoff) cannot be flown during this cycle because, by definition, T_L is the minimal time required to launch an aircraft that is mission-capable.

Event 2

Aircraft that are repaired during the period of length T_L leave maintenance and become mission-capable but are not available for flight during this cycle.

Event 3

All aircraft that were prepared for takeoff are subjected to the probability of ground abort. A ground abort is defined as an unsuccessful attempt by an aircrew to fly an aircraft. The aborted aircraft enter maintenance. No parts demands are generated by ground aborts.

Event 4

The remaining aircraft that were prepared for takeoff fly sorties.

Event 5

Each aircraft that flies is subjected to the probability of attrition and, for each combat loss, an aircraft is deducted from the current strength of the organization.

Event 6

Aircraft that are repaired during the period of length T_F leave maintenance and become mission-capable.

Event 7

Each aircraft returning from flight is subjected to the probability of break, i.e., the probability of requiring essential corrective maintenance prior to flying another combat mission. At the same time, parts demands are generated. Demands that can be filled from stock on-hand result in issues of that stock. Demands that cannot be filled from stock and cannot be satisfied by cannibalization from aircraft that are NMCS result in additional aircraft becoming NMCS.

Event 8

Aircraft that are repaired during the period of length T_W leave maintenance and become mission-capable.

Event 9

This event occurs only after the last flight of the day. It accounts for the parts repair process by subjecting each part in repair to the probability that the repair was completed during the preceding 24 hours. Remaining parts shortages are consolidated on as few aircraft as possible. If the consolidation results in fewer NMCS aircraft than before, the aircraft leaving NMCS status enter maintenance at this time.

Event 10

This event also occurs only after the last flight of the day. Combat losses may be replaced by available reserve aircraft, if the user so specifies. Any remaining reserve aircraft after losses have been replaced are committed according to user specification in the scenario input parameter. If reserves are to be used only as attrition fillers, then any remaining reserve aircraft are left in the reserve pool; thus, the UE for the scenario will never increase. If the user has selected the reserve augmentation mode, then all reserve aircraft will be committed when they become available; hence, the UE for the scenario may actually increase.

THE REPAIR PROCESS

The entry of an aircraft into maintenance results from a ground abort or a "break" during a sortie. In either case, following the ground abort or sortie, the aircraft is subjected to a sequence of random draws that determines the subset of work centers that will be involved in the maintenance on that aircraft. A work center is a set of maintenance personnel with a particular skill. Examples of work centers are the structural repair shop, the hydraulic shop, and the automatic flight control system shop.

In the construction of the maintenance data base that supports the SGM, the aircraft repair times for all work centers involved in the repair of the aircraft are measured from the time of the ground abort or landing of the aircraft. For each work center involved in the repair, a random draw is made from an exponential distribution of repair time for that work center. The mean of that distribution is the reciprocal of the service rate contained in the maintenance data base for the work center in question. All work centers involved in the repair are assumed to work on the aircraft simultaneously; thus, the recovery time of the aircraft is simply the longest of the repair times for all the work centers involved in the recovery of that particular aircraft.

In the SGM, once the aircraft leaves maintenance and becomes mission-capable again, it loses its identity and is counted simply as another aircraft in the mission-capable pool.

2. INPUT DATA

INTRODUCTION

This section provides a description of the inputs required by the Sortie-Generation Model. There are three sets of inputs: user inputs describing the flying scenario and weapon system to be simulated, a maintenance manpower file describing the characteristics of the maintenance work centers, and a recoverable spares file describing the characteristics of the LRUs being modeled. The manpower and spares files are outputs of complex software subsystems described in Volumes V and VI.

SCENARIO DESCRIPTION

At the beginning of each SGM run, the user must specify various inputs which describe the flying scenario to be simulated. The mechanics of this input process are described in Chapter 4. A list of the inputs and sample default values are shown in Figure 2-1. Note that these inputs are divided into two groups: those which are fixed throughout the flying scenario, and those that are allowed to vary by day throughout the scenario. The following is a list of definitions for each of these inputs:

Number of Simulations

The number of independent replications of the simulation experiment with the specified user inputs. The sortie profile is computed as a simple average of these replications.

Random Number Seed

A number used to initialize the pseudo-random number generator used in the simulation. Changing this value produces a different random number stream and allows the user to conduct independent experiments. The user can reproduce a previous experiment by using the same seed and the same input files.

```
THE CURRENT VALUES OF THE SCENARIO INPUTS ARE :

 INPUT          SCENARIO ITEM             CURRENT
 CODE                                      VALUE

   1    # SIMULATIONS                    =  40
   2    RANDOM NUMBER SEED               =  12.3
   3    UE                               =  72
   4    AIRCRAFT BREAK RATE              =  .20
   5    INITIAL NMCM RATE                =  .3
   6    # DAYS                           =  30
   7    FIRST TAKEOFF TIME               =  0600
   8    LAST TAKEOFF TIME                =  1824
   9    SORTIE LENGTH (HRS)              =  1.7
  10    MINIMAL RECOVERY TIME (HRS)      =  1.4
  11    INFINITE MANPOWER (YES/NO)       =  NO
  12    INFINITE SPARE PARTS (YES/NO)    =  NO
  13    AUGMENT RESERVE AC (YES/NO)      =  NO
  14    MAX SORTIES/DAY FOR PLOT(OR 0)   =  0

THE FOLLOWING ITEMS MAY VARY BY DAY(D) OR CYCLE/DAY(C/D)

  15    ATTRITION RATE           (D)     =  .0
  16    GROUND ABORT RATE        (D)     =  .04
  17    # MASS LAUNCHES PER DAY  (D)     =  5
  18    RESERVE AIRCRAFT         (D)     =  0
  19    MAXIMUM LAUNCH-SIZE     (C/D)    =  72
```

FIGURE 2-1

SCENARIO INPUTS WITH SAMPLE DEFAULT VALUES

UE (Unit Equipment)

The initial number of possessed aircraft at the base of interest. This number does not include the available reserve aircraft.

Aircraft Break Rate

Probability that an aircraft returning from a sortie requires unscheduled maintenance in one or more work centers prior to further flight.

Initial NMCM Rate

Proportion of the possessed aircraft that are not-mission-capable-maintenance at the start of the flying scenario. For example, if the user specifies an initial NMCM rate of 0.3 with a UE of 72, the SGM will begin each simulation experiment with 22 aircraft undergoing maintenance. The remaining

50 aircraft will initially be either mission-capable or waiting for a recoverable spare part.

<u>Number of Days</u>

The number of flying days in the scenario of interest.

<u>First and Last Takeoff Times</u>

Scheduled takeoff times for the first and last mass launches of the day. These times are specified using 24-hour clock time, e.g., 1:00 o'clock p.m. would be input as 1300 hours. The remaining scheduled mass launches are equally spaced between the first and last takeoff times.

<u>Sortie Length</u>

Fixed length in hours of each sortie.

<u>Minimal Recovery Time</u>

The minimal required time between the landing of the aircraft and takeoff for the next sortie, given that no corrective maintenance is required. It includes only the time required to taxi, park, chock, shut down, refuel, rearm, inspect, restart, and launch.

<u>Infinite Manpower</u>

This input allows the user to generate a sortie profile with or without maintenance manpower constraints. (See MAINTENANCE MANPOWER, page 2-5.) If the user inputs YES as the current value of scenario item 16, INFINITE MANPOWER (YES/NO), the SGM assumes that there are always sufficient maintenance teams in every work center so that no aircraft ever has to wait in a queue to begin unscheduled maintenance. Thus, the SGM ignores the number of maintenance teams in the maintenance manpower input file and uses an infinite server assumption. The work center break rates and service rates still apply. If the user inputs NO, the SGM uses the number of maintenance teams in the input file as a constraint on maintenance capacity.

Infinite Spare Parts

This input allows the user to generate a sortie profile with or without spares constraints. (See RECOVERABLE SPARE PARTS, page 2-9.) If the user inputs YES as the current value of scenario item 17, INFINITE SPARE PARTS (YES/NO), the SGM assumes that there are always sufficient spare parts, i.e., that throughout this flying scenario, no aircraft is NMCS. Thus, no parts demands or repairs are simulated in this run. If the user inputs NO, the SGM uses the asset data from the spares input file as a constraint on inventory.

Augment Reserve Aircraft

This input allows the user to specify how the reserve aircraft are to be committed. If the user inputs YES, then all reserve aircraft are committed on the day they become available; hence, the UE-size of the force may actually increase during the scenario. If NO is input, then the reserves are used only as attrition fillers to replace combat losses; thus, not all reserves may be committed on the day they become available.

Maximum Sorties/Day For Plot

This parameter is used to set the maximum vertical scale on the sorties-per-day plot of the SGM results. For example, if the user wanted plots of a series of SGM runs, he would use this parameter to ensure all the plots are on the same scale. If zero is input, the scale is determined from the maximum sorties per day that occur in the SGM results.

Attrition Rate

Probability that an aircraft does not return from a sortie due to combat attrition. This rate may be different for each day of the scenario.

Ground-Abort Rate

Probability that an aircraft undergoes some failure right before takeoff that requires unscheduled maintenance that renders it not-mission-capable. This rate may be different for each day of the scenario.

Number of Mass Launches Per Day

The number of equally spaced flying periods for each day in the scenario. This number is allowed to vary for each day of the scenario.

Reserve Aircraft

The number of aircraft in reserve that are available to augment the current UE of the scenario. As described previously, the user may specify whether these reserves are committed on the day they become available or are to be used only as attrition fillers to replace combat losses. The number of reserve aircraft arriving on the scene may be specified for each day of the scenario.

Maximum Launch Size

The maximum number of aircraft to be scheduled on any particular wave. The default value normally is equal to the UE value. This default is used because of the maximal surge scenario of the SGM; it forces all mission-capable aircraft to fly each period.

MAINTENANCE MANPOWER

The SGM represents aircraft maintenance as a queueing process. When an aircraft breaks (malfunctions and cannot take off for the next scheduled sortie), on-aircraft unscheduled maintenance is performed by independent, parallel work centers. Each work center consists of individuals with a specific skill type. Skill type is defined by an individual's Air Force Specialty Code (AFSC). A work center, besides possessing individuals with a specific skill type, is represented by three other characteristics. These characteristics (break rate, number of servers, and service rate) are used to represent the work center in the queueing model.

The Sortie-Generation Model system consists of the SGM, a spares subsystem, and a maintenance subsystem. The purpose of the maintenance subsystem is to translate standard Air Force maintenance performance and authorization data into the queueing model inputs just described. Generally, the modified Common Data Extraction Program (CDEP) portion of the maintenance subsystem selects records (tasks) based on the type of maintenance and the performing work center (AFSC). The Maintenance Manpower and Performance Analyzer (MMPA) then reads the task data, AFSC authorizations from the unit manning document, and the number of sorties flown during the same time period to which the maintenance data apply. After reading the data, the MMPA combines tasks into jobs and manipulates the job data along with the number of sorties and manpower authorizations to produce the work center (AFSC) data required by the SGM.

Use of CDEP requires specification of the workcenters (AFSCs) of interest. The decision of what workcenters to include (model explicitly in the SGM) depends on the work center's function and how that function coincides with the SGM's use as a budgeting tool. Aircraft maintenance is performed by four work center types:

1. Those work centers assigned to the Aircraft Generation Squadron (AGS) which perform unscheduled on-aircraft maintenance.

2. Those work centers in the Component Repair Squadron (CRS) or Equipment Maintenance Squadron (EMS) which perform unscheduled on-aircraft maintenance as a primary function or in support of the AGS when needed.

3. CRS and EMS work centers whose primary function is performance of scheduled or off-equipment maintenance.

4. Those CRS and EMS work centers whose on-aircraft unscheduled maintenance can be deferred.

Examples of the four work center types are:

1. 325X0 - Automatic Flight Control System
   325X1 - Instrument Systems

2. 423X0 - Electrical System
   423X1 - Environmental System
   423E3 - Fuel System
   423E2 - Egress System

3. 426R2 - Jet Engine Intermediate Maintenance Shop
   431R1 - Inspection Section

4. 427R1 - Corrosion Control

Only work center types 1 and 2 are explicitly modelled in the SGM. Work center types 3 and 4 are not modelled because we assume that most scheduled and deferable maintenance will not be performed during a maximal surge effort.

Break Rates

Two types of break rates are used in the SGM. They are an aircraft break rate and individual work center break rates. The aircraft break rate is specified by the user in specifying the scenario and represents an estimate of the proportion of sorties flown in surge exercises that result in the need for essential corrective maintenance prior to further flight. A surge estimate is used because it reflects the purpose of the SGM; i.e., estimation of maximal wartime sortie capability. In other words, the surge break rate incorporates the tendency for pilots to hip-pocket (not report) non-grounding failures until after the last sortie of the day. The result is a lower break rate than reported during normal peacetime (training) operations.

A sample F-4E input file for Seymour Johnson AFB is shown in Figure 2-2. The reader should refer to this figure when reading the following work center input definitions.

Given a broken aircraft, the work center break rate reflects the probability that corrective maintenance is required in a specific work center

```
*********************************************
**********  AIRCRAFT MAINTENANCE  **********
*********************************************

                    BREAK     TOTAL    SERVICE RATE
WC #     AFSC       RATE     SERVERS   (ACFT/HOUR)

  1     321X2     0.2878     27.77        0.1417
  2     325X0     0.1515     15.06        0.1384
  3     328R3     0.1062     31.07        0.1273
  4     328X0     0.2010     18.36        0.1769
  5     328X4     0.1506      9.57        0.2507
  6     404X1     0.0225     12.00        0.1510
  7     423E2     0.1699      9.95        0.0682
  8     423E3     0.0608      8.31        0.1043
  9     423X0     0.1188     12.28        0.1327
 10     423X1     0.0793      6.57        0.1571
 11     423X4     0.0836     11.91        0.1365
 12     426X2     0.0508     10.81        0.1585
 13     427R0     0.0379      4.56        0.3955
 14     427X5     0.1633     14.89        0.2584
 15     431E1     0.0335     10.73        0.0857
 16     431X1     0.0527    131.49        0.5356
 17     462X0     0.1641      7.27        0.5434
```

FIGURE 2-2

SAMPLE MANPOWER INPUT FILE

(SEYMOUR JOHNSON AFB, 1980: F-4E)

prior to further flight. Maintenance can be required in more than one work center at the same time.

Estimation of the work center break rate is based on the probability that one or more jobs (on the same aircraft) require corrective maintenance in the same work center following a sortie. A job is a collection of related corrective tasks. For example, a job may consist of the following types of tasks: troubleshoot the break, remove and replace the failed item, and verify operation of the new item.

Work Center Service Rates

The work center service rate is the rate, in aircraft per hour, unscheduled on-aircraft maintenance is performed by a maintenance team (defined later in detail). The service rate is the inverse of the expected service time which represents the expected time required to complete all jobs on an aircraft which has broken into the work center. The job time begins as soon as the broken aircraft lands or ground aborts, and includes the time to perform the individual tasks as well as the idle time between tasks. The job time ends when the last task in the job is completed.

Number of Servers

The number of servers represents the number of maintenance teams in a work center that are available to perform unscheduled on-aircraft maintenance. A server, or maintenance team, is composed of one or more individuals. Calculation of the number of servers is given by equation (1).

$$\text{Number of Servers} = \frac{\text{Funded Authorizations}}{\begin{array}{c}\text{Expected Number of Men Working}\\ \text{On The Aircraft}\end{array}} \qquad (1)$$

Funded authorizations for a work center (AFSC) are obtained from the Unit Manning Document. Only type 1 and 2 work centers, as described earlier, are included in counting the authorizations for an AFSC. The CRS and AGS authorizations are added together when the same AFSCs are found in both squadrons. This total gives the number of individuals available to perform unscheduled on-aircraft maintenance in a surge environment. The number of men working on the aircraft is an expected value that accounts for the possibility of more than one job being worked on (i.e., more than one crew at work) in a work center at once.

The number of servers calculated by equation (1) and shown in Figure 2-2 is divided by two in the SGM to give the number of servers used in the queueing model. Division by two reflects the availability of people twelve hours per day. Thus, we assume two twelve-hour shifts, each with the same number of people.

RECOVERABLE SPARE PARTS

In order to simulate the failure and repair of recoverable LRUs and their impact on sortie-generation capability, the SGM needs, for each LRU that is installed on the aircraft of interest:

a. The configuration of the component's installation on the aircraft

b. The removal rate

c. The asset position at the start of the scenario

d. The average base resupply time and the average depot resupply time

e. The BNRTS (Base Not Repairable This Station) percentage

The installation configuration, the BNRTS percentage, and the removal rate are obtained directly from the Air Force D041 data base. This data base contains data for all aircraft recoverable parts in the Air Force inventory. Maintained and updated quarterly by the Air Force Logistics Command, this data base is typically used for making budget projections and requirements computations.

The starting asset position, the average base repair time, and the average depot resupply time are computed by the LMI Aircraft Availability Model. The budget levels for procurement and repair of components must be established by the user through the Interactive Budget Allocation Program prior to running the SGM. A partial listing of a recoverable-spares input file is shown in Figure 2-3.

NSN	REMOVAL RATE	QPA	FAP	INITIAL STOCK	INITIAL NO. IN RESUPPLY	BASE NRTS	RESUPPLY TIMES (DAYS) BASE	RESUPPLY TIMES (DAYS) DEPOT
1430010454699BF	.01786	1	1.0	5	8.156	0.08	6.0	26.2
1430010387038BF	.01700	1	1.0	6	7.211	0.05	6.0	27.4
2620000884523	.01708	2	0.9	90	28.649	1.00	0.	18.0
5865001994210EW	.00109	4	1.0	1	1.405	0.90	6.0	31.0
1430010399244BF	.01285	1	0.3	1	1.744	0.08	6.0	27.7
6610004629837BF	.00404	1	1.0	6	3.091	0.85	5.0	14.7
1430010610350BF	.00607	1	1.0	3	2.226	0.	6.0	0.
1630004463778	.01676	2	1.0	33	9.226	0.06	5.0	14.0
1270010588980	.00483	1	1.0	4	0.937	0.45	6.0	11.5
5826010395000	.00600	1	0.3	1	1.373	0.30	9.0	22.1
5826010401785	.00692	1	0.3	2	1.650	0.40	8.0	21.0
1430010387055BF	.00484	1	1.0	3	1.742	0.	6.0	0.
5826010183511	.00168	2	1.0	9	3.878	0.60	4.0	34.6
1430002356325BF	.01023	1	0.9	6	2.282	0.09	3.0	17.1
2840008717414PL	.00084	2	1.0	2	2.431	0.76	6.0	28.6
5865000233292EW	.00200	1	1.0	0	0.005	1.00	0.	6.8
5865003713344EW	.00133	4	1.0	3	0.723	0.97	3.0	13.3
1270000641997	.00441	1	1.0	6	1.345	0.66	4.0	10.9
6115008681999EW	.00184	5	0.2	1	0.092	0.43	2.0	78.8
5865000999348EW	.00103	5	1.0	2	0.548	0.98	2.0	12.3

FIGURE 2-3

SAMPLE RECOVERABLE SPARES INPUT

Installation Configuration

Not all components are installed one per aircraft. On some aircraft in which a component is used, more than one may be installed. Furthermore, not every aircraft of a given type will be in a standard configuration. Two elements of input data are used to define a component's installation configuration, the quantity per aircraft (QPA) and the future application percentage (FAP). The QPA defines the component's standard configuration with respect to a specific model/design/series aircraft. The FAP is used for a variety of purposes in the D041 system but is interpreted by the SGM to denote the proportion of aircraft with the component installed in standard configuration. Thus, component flying hours may be computed by multiplying aircraft flying hours times QPA times FAP.

Removal Rate

The component removal rate is defined as the expected number of component removals per component flying hour. It is computed in a straightforward manner using the daily demand rate (DDR) from the D041 data base and the peacetime flying hours for all the aircraft types which use the component. An important assumption of the SGM system is that the expected number of component removals is directly proportional to the number of flying hours flown.

Starting Asset Position

The starting asset position for a component is defined by the expected number in resupply at the start of the scenario and the authorized number of spares. By resupply we mean base repair, depot repair, or in transit. If the authorized number of spares is greater than the starting number in resupply, then there will be spares on hand at the start of the scenario. If the authorized number of spares is less than the starting number in resupply, then there will be unsatisfied (backordered) demands at the start of the scenario. The starting number in resupply is determined independently for each replication run by the SGM. The model does this by drawing at random a starting number from a Poisson probability distribution whose mean is the expected number in resupply.

Resupply Times

The average base resupply time is the average number of days that a component is in the base repair shop before it is returned to servicable status. This equals the shop repair time plus the average time awaiting SRUs (if any). The average depot resupply time is the average number of days between when an order for a component is made to the depot and when the component is received at the base. This equals the order and ship time plus the average delay at the depot awaiting an available spare to ship.

BNRTS Percentage

The percentage of demands which are not base repairable. A demand which is not base repairable may be condemned or repaired at the depot but in either case an order will be placed for depot resupply.

The spares subsystem excludes many components from the SGM inputs on the basis that they will not have any impact on the estimated sortie-generation capability. Components are excluded if their removal rates are less than .0005 per hour. All components whose demands are not flying-hour dependent are also excluded.

3. OUTPUTS

INTRODUCTION

The Sortie-Generation Model outputs consist of a summary of the user-specified scenario, a sortie profile showing the average number of sorties flown for each period of the scenario, and a graph of the daily sortie production.

SCENARIO SUMMARY

A summary of the user inputs describing the scenario being simulated is printed at the beginning of each SGM run. A sample of this summary is shown in Figures 3-1 and 3-2. The first section describes dimensions of the

```
*************************************************************
*************************  SGM RUN  *************************
*************************************************************

SIMULATION -   REPLICATIONS =  40     RANDOM NUMBER SEED = 12.3
AIRCRAFT -   UE = 72    RESERVES = 24    MAXIMUM LAUNCH-SIZE = 72


FLYING SCHEDULE -

         WAVES   TAKEOFF TIMES     MINIMAL    SORTIE  WAIT  OVERNIGHT
DAYS   PER DAY   FIRST   LAST    TURNAROUND   LENGTH  TIME  RECOVERY

 30       5      0600    1824       1.40       1.70   0.00    8.50
RATES -

   INITIAL                    AIRCRAFT
  NMCM RATE   ATTRITION      BREAK RATE   GROUND-ABORT

    0.150       0.01          0.2000       0.0400


LRU TYPES -  262
```

FIGURE 3-1

SCENARIO SUMMARY

```
*******************************************
**********  AIRCRAFT MAINTENANCE  **********
*******************************************

                   BREAK    TOTAL    SERVICE RATE
 WC #    AFSC      RATE     SERVERS  (ACFT/HOUR)

   1     321X2    0.2878    27.77       0.1417
   2     325X0    0.1515    15.06       0.1384
   3     328R3    0.1062    31.07       0.1273
   4     328X0    0.2010    18.36       0.1769
   5     328X4    0.1506     9.57       0.2507
   6     404X1    0.0225    12.00       0.1510
   7     423E2    0.1699     9.95       0.0682
   8     423E3    0.0608     8.31       0.1043
   9     423X0    0.1188    12.28       0.1327
  10     423X1    0.0793     6.57       0.1571
  11     423X4    0.0836    11.91       0.1365
  12     426X2    0.0508    10.81       0.1585
  13     427R0    0.0379     4.56       0.3955
  14     427X5    0.1633    14.89       0.2584
  15     431E1    0.0335    10.73       0.0857
  16     431X1    0.0527   131.49       0.5356
  17     462X0    0.1641     7.27       0.5434
```

FIGURE 3-2

WORK CENTER SUMMARY

simulation, numbers of aircraft, flying schedule, and various rates and probabilities. The second section lists the work-center characteristics provided in the maintenance-manpower input file. Definitions of these scenario and work-center inputs are provided in Chapter 2.

SORTIE PROFILE

The sortie profile is a summary of the average numbers of aircraft in the various states modeled by the SGM: Mission-capable, Maintenance, NMCS, Combat Loss, or Reserve. The numbers of aircraft in each SGM state are collected at the beginning of each sortie period for each flying day, and an average is computed from the total for all simulation replications. A sample of a sortie profile is shown in Figure 3-3. A short description of the information provided by each output column follows:

DAY	PER	SORTIES/ PERIOD	SORTIES/ DAY	SORTIES/ AC	NMCM	NMCS	CUM. LOSSES	RES. LEFT
1	1	55.3			10.0	4.6	0.	24.0
	2	45.3			18.7	5.9	0.5	
	3	38.7			23.4	7.4	0.9	
	4	34.9			25.8	8.6	1.3	
	5	31.8	206.1	2.90	27.4	9.7	1.6	
			206.1					
2	1	48.1			13.4	9.0	1.9	22.1
	2	40.5			19.3	10.3	2.3	
	3	35.0			23.2	11.6	2.6	
	4	31.9			24.8	12.7	2.9	
	5	29.8	185.3	2.60	25.8	13.9	3.2	
			391.4					
3	1	45.0			12.9	12.3	3.5	20.5
	2	37.8			19.0	13.5	3.7	
	3	33.1			22.3	14.8	4.0	
	4	30.0			24.4	15.9	4.1	
	5	28.6	174.4	2.44	24.5	16.8	4.4	
			565.8					
4	1	42.3			13.2	14.8	4.7	19.3
	2	35.8			18.2	16.2	5.1	
	3	30.8			21.6	17.5	5.6	
	4	28.0			23.2	18.4	5.9	
	5	26.4	163.4	2.29	23.5	19.4	6.1	
			729.2					
5	1	41.2			12.2	17.0	6.3	17.8
	2	34.1			17.7	18.5	6.7	
	3	30.7			20.1	19.3	7.2	
	4	28.4			21.0	20.2	7.7	
	5	26.5	160.8	2.26	22.0	20.7	8.0	
			890.0					

FIGURE 3-3
SGM SORTIE PROFILE
(FIRST FIVE DAYS)

Day and Period

The DAY and PER columns indicate the flying day and period, respectively, for which the averages were collected.

Sorties per Period

The SORTIES/PERIOD column indicates the average number of sorties flown for this day and period.

Sorties per Day

The SORTIES/DAY column reflects the total number of sorties flown on each flying day. A cumulative sortie total for all days thus far is also printed immediately below the total for each day.

Sorties per Aircraft

The SORTIES/AC column provides an indication of the average number of sorties being flown by each on-the-scene aircraft for this particular flying day. An on-the-scene aircraft is defined as an aircraft which is either mission-capable, in maintenance, or NMCS; i.e., it is not in the COMBAT LOSS or RESERVE state. This statistic is computed by dividing the average number of sorties flown each day by the average number of on-the-scene aircraft for that day. This on-the-scene average is computed by totaling the number of on-the-scene aircraft at the start of each flying period for the day.

Maintenance

The NMCM column reflects the average number of aircraft that were unable to fly each period because they were in maintenance at the start of the pre-takeoff period. This pre-takeoff period begins at a time T_L before scheduled takeoff (see Figure 1-3). This average includes any aircraft which may have been repaired during the pre-takeoff period and it does not include any aicraft which may have ground-aborted before takeoff.

NMCS

This is the average number of aircraft that were unable to fly each period because they were in the NMCS state at the start of the pre-takeoff period.

Cumulative Combat Losses

The CUM. LOSSES column indicates the cumulative number of combat losses at the beginning of this flying period.

Remaining Reserves

The RES. LEFT column is the average number of reserve aircraft remaining at the start of each flying day. Since reserve aircraft are committed only once at the end of each flying day to replace combat losses, this average is only printed for this first flying period of the day to indicate the remaining reserves.

SORTIE PLOTS

In addition to the sortie profile for each SGM run, graphs of the average sorties per aircraft per day and average sorties flown are printed. Samples of these plots are shown in Figures 3-4 and 3-5.

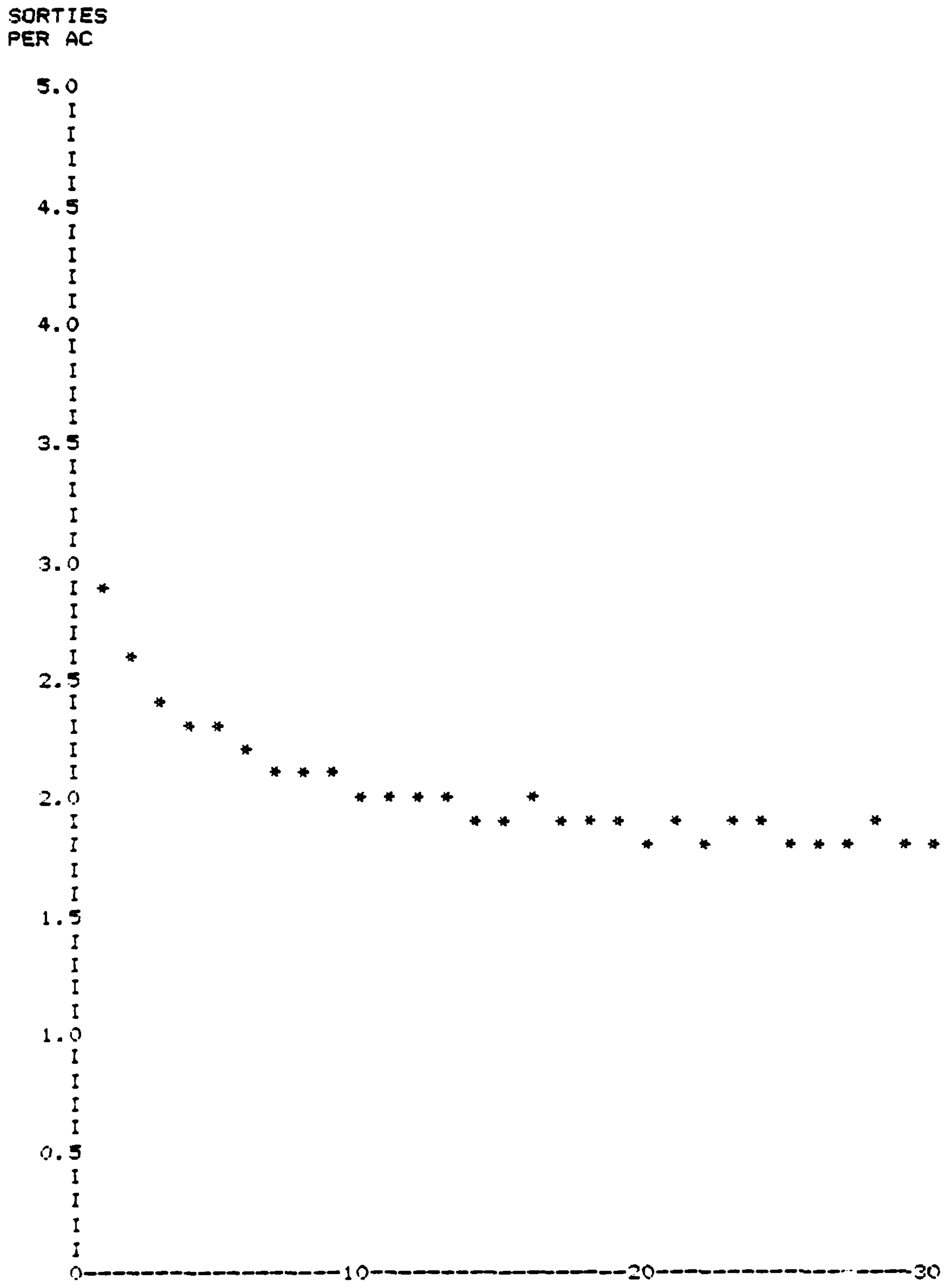


FIGURE 3-4

SGM SORTIE PLOT: SORTIE RATES

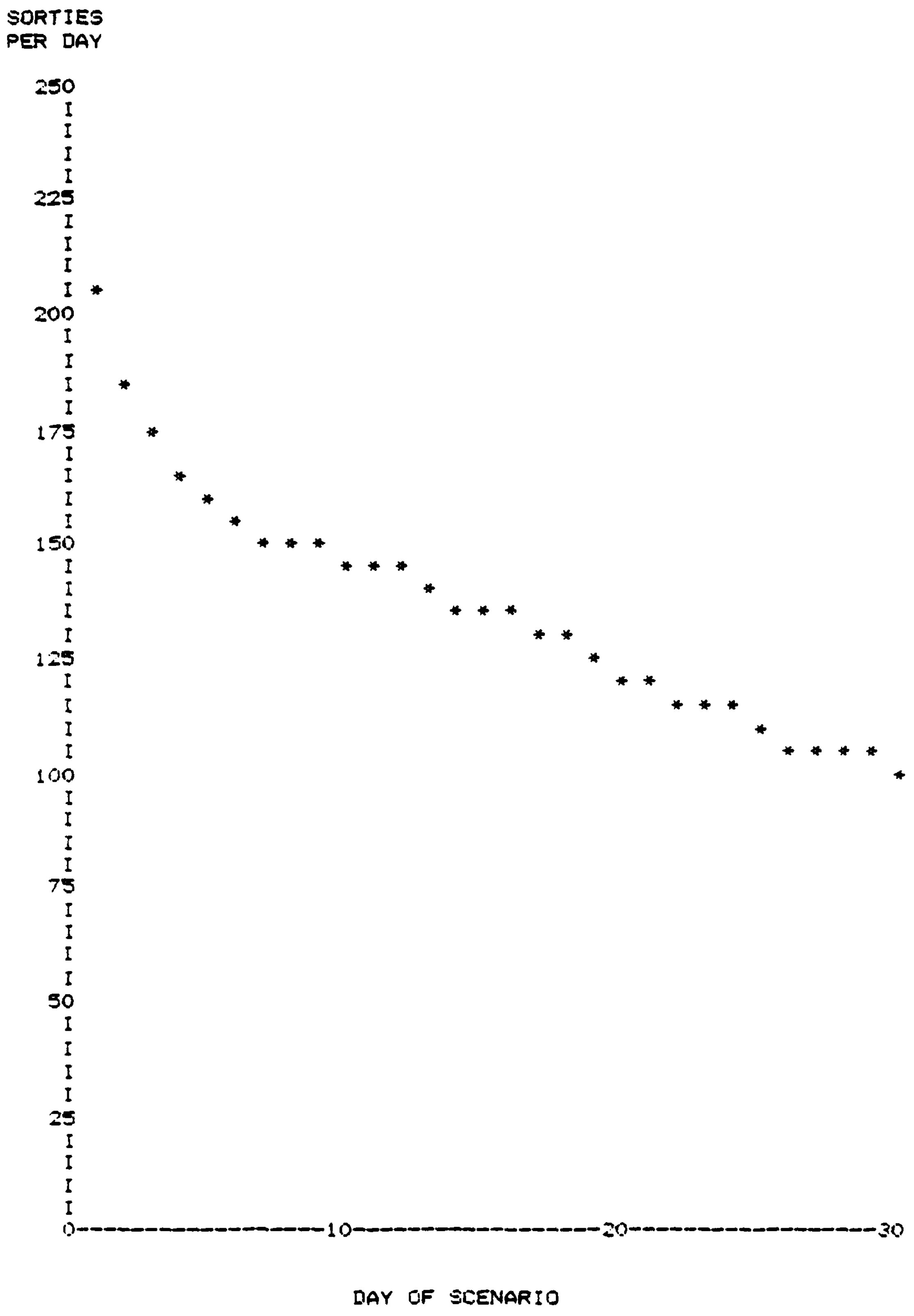


FIGURE 3-5

SGM SORTIE PLOT: SORTIES PER DAY

4. RUN INSTRUCTIONS

ENVIRONMENT

The Sortie-Generation Model (SGM) has been developed on System C, an unclassified computer system located at the Pentagon and supported by the Air Force Data Services Center (AFDSC). This system operates on a Honeywell G-635 computer under the series 600/6000 GCOS Time-Sharing System. Access to the system is possible on remote terminals by a dial-up procedure.

The SGM has been written in the Honeywell 600/6000 FORTRAN programming language, the only version of FORTRAN available on the system. The run process has been designed so that the model may be run in either the remote-batch or time-sharing modes. There are advantages and disadvantages to both procedures. If System C is carrying a light load (i.e., only a few users are signed on), then a time sharing run is significantly faster; however, throughout the simulation the terminal cannot be used for any other purpose and it is not possible to direct the output elsewhere. Once a job has been submitted interactively to be run as a batch job, the user is free to make other runs, use the terminal for some other purpose, or even to log-off the computer.

For a detailed description of System C, the Time-Sharing System, and FORTRAN 600/6000, the user is referred to the following manuals:

a. AFDSC User's Handbook, Volume III - G-635 Computer Systems

b. Honeywell Series 600/6000 Time-Sharing System General Information Manual

c. Honeywell Series 600/6000 FORTRAN Manual

RUN STEPS

There are three steps in obtaining an SGM run (see Figure 4-1): initiate communication with the Time-Sharing System (log-on System C); specify scenario

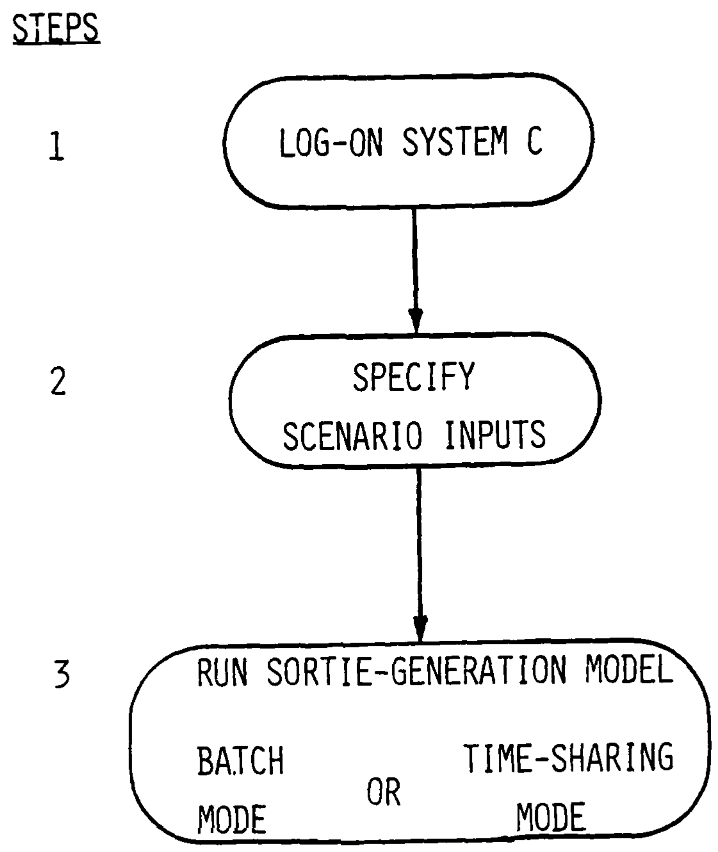


FIGURE 4-1
RUNNING THE SGM

inputs for the run; and initiate execution of the SGM in either the remote-batch or time-sharing mode. Figure 4-2 is a list of the exact commands to be typed by the user for a batch or time-sharing run. Appendix A provides listings of the Job Control Language (JCL) and Run-Command files which are initiated by these listed commands.

TIME-SHARING RUN COMMAND SEQUENCE

```
            - - -  LOG-ON SYSTEM C  - - -
RUNY LA61A/SUBMIT,R
RUNC OS29/N232D/SGM/SETPARAM
            - - - SET SCENARIO PARAMETERS - - -
RUNC OS29/N232D/SGM/RSGMTSS
```

BATCH RUN COMMAND SEQUENCE

```
            - - -  LOG-ON SYSTEM C  - - -
RUNY LA61A/SUBMIT,R
RUNC OS29/N232D/SGM/SETPARAM
            - - - SET SCENARIO PARAMETERS - - -
RUN OS29/N232D/SGM/RSGMBTCH
```

FIGURE 4-2
SGM RUN INSTRUCTIONS

The remainder of this section provides a description of each step of the SGM run process.

Log-on System C

To initiate communication with the System C Time-Sharing System, the user must connect an appropriate remote terminal to System C and enter a unique user identification code, a verification password, and the appropriate project identification code. These codes are assigned by AFDSC. A complete description of the log-on procedures for the various types of remote terminals is provided in the Honeywell Series 600/6000 Time-Sharing System General Information Manual referenced previously.

Specifying Scenario Inputs

The next step for either a batch or time-sharing SGM run is to specify the scenario. The scenario inputs are defined in Chapter 2 of this guide. Entry of the scenario parameters is performed via an interactive program with a question-and-answer format. The following commands initiate this interactive program:

```
RUNY LA61A/SUBMIT,R
RUNC OS29/N232D/SGM/SETPARAM
```

The user is immediately prompted for the aircraft type which is used to select the appropriate file containing the default scenario values for that aircraft. (Default files currently exist for the following aircraft types: A-10, F-4, F-15, F-16, and F-111.) Next, the user is asked for a random-number seed to initialize the random-number generator. Then, the default scenario parameters are loaded, and the user is allowed to list or change these scenario parameters. If any changes are desired; the user specifies a parameter to be changed and the corresponding new value. This process continues until the user has made all the desired changes; the scenario program is then terminated and the SGM can be run in either a batch or time-sharing mode. Figure 4-3 provides an example of an interactive scenario-input session. The user responses for this session are underlined and numbered, and an explanation of each response is provided in Figure 4-4.

Time-Sharing Run

To run the SGM in this mode, the user types: RUNC OS29/N232D/SGM/RSGMTSS. This program asks the user for the maintenance-manpower file and recoverable-spares file (see Chapter 2 for description of these files). The SGM then begins execution and the results (see Chapter 3 for description of outputs) are printed at the user's remote terminal.

Batch Run

To run in batch mode, the user types: RUN OS29/N232D/SGM/RSGMBTCH. The program asks for the maintenance-manpower file and recoverable-spares file as described above. After the job is submitted to System C batch, a five-digit identification number is printed to enable the user to track the job while it is being processed. The SGM output is directed to the printer station specified by the JCL run file (currently set for the LMI remote printer).

Log-off Procedure

Termination of an SGM *run session* is a two-step procedure. First, the user types DONE when prompted by an "=" after the end of the SGM run; this terminates the interactive submission program. Then, the user types BYE to disconnect the terminal from System C Time-Sharing.

```
(1)  SYSTEM ?RUNY LA61A/SUBMIT,R

     ***** STARS SUBMIT SUBSYSTEM *****

(2)  =RUNC OS29/N232D/SGM/SETPARAM
     ENTER AC-TYPE      ?
(3)  =F4

      ENTER RANDOM NUMBER SEED
(4)  =12.3

      CODE - FUNCTION
         1   SET PARAMETERS FOR SGM RUN
         2   LIST CURRENT SCENARIO
         3   CHANGE SCENARIO

      ENTER FUNCTION CODE (1-SET/2-LIST/3-CHANGE)
(5)  =2

     THE CURRENT VALUES OF THE SCENARIO INPUTS ARE :

      INPUT           SCENARIO ITEM              CURRENT
      CODE                                        VALUE

        1    # SIMULATIONS                     =  40
        2    RANDOM NUMBER SEED                =  12.3
        3    UE                                =  72
        4    AIRCRAFT BREAK RATE               =  .20
        5    INITIAL NMCM RATE                 =  .3
        6    # DAYS                            =  30
        7    FIRST TAKEOFF TIME                =  0600
        8    LAST TAKEOFF TIME                 =  1824
        9    SORTIE LENGTH (HRS)               =  1.7
       10    MINIMAL RECOVERY TIME (HRS)       =  1.4
       11    INFINITE MANPOWER (YES/NO)        =  NO
       12    INFINITE SPARE PARTS (YES/NO)     =  NO
       13    AUGMENT RESERVE AC (YES/NO)       =  NO
       14    MAX SORTIES/DAY FOR PLOT(OR 0)    =  0

     THE FOLLOWING ITEMS MAY VARY BY DAY(D) OR CYCLE/DAY(C/D)

       15    ATTRITION RATE           (D)      =  .0
       16    GROUND ABORT RATE        (D)      =  .04
       17    # MASS LAUNCHES PER DAY  (D)      =  5
       18    RESERVE AIRCRAFT         (D)      =  0
       19    MAXIMUM LAUNCH-SIZE     (C/D)     =  72

      ENTER FUNCTION CODE (1-SET/2-LIST/3-CHANGE)
(6)  =3

      IF CHANGES ARE DESIRED ENTER THE INPUT CODE OF THE ITEM
      TO BE ALTERED, ELSE ENTER 0
(7)  =5
```

Circled Numbers Indicate User Responses

```
      ENTER NEW VALUE OF ITEM, OLD VALUE=.3
(8)  =.15

      ENTER NEXT INPUT CODE OR 0 IF FINISHED
(9)  =18

      DO YOU WANT PARAMETER 18  TO VARY BY DAY ?
(10) =NO

      ENTER NEW VALUE OF ITEM, OLD VALUE=0
(11) =24

      ENTER NEXT INPUT CODE OR 0 IF FINISHED
(12) =0

      ENTER FUNCTION CODE (1-SET/2-LIST/3-CHANGE)
(13) =2

     THE CURRENT VALUES OF THE SCENARIO INPUTS ARE :

      INPUT           SCENARIO ITEM              CURRENT
      CODE                                        VALUE

        1    # SIMULATIONS                     =  40
        2    RANDOM NUMBER SEED                =  12.3
        3    UE                                =  72
        4    AIRCRAFT BREAK RATE               =  .20
        5    INITIAL NMCM RATE                 =  .15
        6    # DAYS                            =  30
        7    FIRST TAKEOFF TIME                =  0600
        8    LAST TAKEOFF TIME                 =  1824
        9    SORTIE LENGTH (HRS)               =  1.7
       10    MINIMAL RECOVERY TIME (HRS)       =  1.4
       11    INFINITE MANPOWER (YES/NO)        =  NO
       12    INFINITE SPARE PARTS (YES/NO)     =  NO
       13    AUGMENT RESERVE AC (YES/NO)       =  NO
       14    MAX SORTIES/DAY FOR PLOT(OR 0)    =  0

     THE FOLLOWING ITEMS MAY VARY BY DAY(D) OR CYCLE/DAY(C/D)

       15    ATTRITION RATE           (D)      =  .0
       16    GROUND ABORT RATE        (D)      =  .04
       17    # MASS LAUNCHES PER DAY  (D)      =  5
       18    RESERVE AIRCRAFT         (D)      =  24
       19    MAXIMUM LAUNCH-SIZE     (C/D)     =  72

      ENTER FUNCTION CODE (1-SET/2-LIST/3-CHANGE)
(14) =1

      TO BEGIN SIMULATION USE EITHER,

         RUNC OS29/N232D/SGM/RSGMTSS - FOR TIME SHARING RUN
      OR,
         RUN OS29/N232D/SGM/RSGMBTCH - FOR BATCH RUN.
```

FIGURE 4-3. SAMPLE INTERACTIVE SESSION -- SCENARIO INPUTS

User Response	Action	User Response	Action
1	Execute Under LMI SUBMIT Subsystem.	7 - 8	Change Initial NMCM Rate Parameter (#5); Set New Value to .15.
2	Begin Execution of Scenario-Input Program	9 - 11	Change Reserve-Aircraft Parameter (#18); Set New Number of Reserves to 24 Aircraft.
3	Enter Aircraft Type	12	Indicate No More Changes Desired At This Time
4	Enter Random-Number Seed	13	List Current Scenario (With New Changes).
5	List Current Scenario Values.	14	Terminate Scenario-Input Session-Scenario Values Are Now Set.
6	Initiate Change Scenario Values Option.		

FIGURE 4-4

EXPLANATION OF SCENARIO-INPUT SESSION

APPENDIX A

SGM JOB CONTROL LANGUAGE (JCL)

This appendix provides listings of the JCL and Run-Command files used to run the Sortie-Generation Model (SGM):

OS29/N232D/SGM/RSGMBTCH

This is the JCL file for running the SGM in the Batch mode. Output is directed to remote printer at LMI.

OS29/N232D/SGM/RSGMTSS

This is the Run-Command file for running the SGM in the Time-Sharing mode under the LMI SUBMIT subsystem.

OS29/N232D/SGM/SETPARAM

This is the Run-Command file for setting the scenario parameters (in Time-Sharing mode) under the LMI SUBMIT subsystem.

OS29/N232D/SGM/ZD.F4

This is the file which provides the default values for initializing scenario parameters for the F-4.

```
     0S29/N232D/SGM/RSGMBTCH

100##S,R(XL) :,8,16,58
110$:NOTE:** MIKE **  0S29/N232D/SGM/RSGMBTCH
120$:IDENT:0S2011N241D ,0S29UGOODWIN
130$:OPTION:FORTRAN,NOMAP
140$:SELECT:0S29/N232D/SGM/CSGM
150$:EXECUTE
160$:LIMITS:14,27K,,1K
170$:DATA:01,NCKSUM,COPY
180$:SELECTD:0S29/N232D/SGM/PARAMS
190$:ENDCOPY
200$:PRMFL:02,R,S,0S29/N241D/CDEP/SGMINPT2/&MANPOWERBASE.
210$:PRMFL:04,R,S,LA61A/SLAY/DATA/&AC-TYPE./&SPARESFILE.
220$:FILE:03,A3S
230$:FILE:07,A1S
240$:OPTION:FORTRAN,NOMAP
250$:SELECT:0S29/N232D/SGM/CPLOT
260$:EXECUTE
270$:LIMITS:1,13K,,2K
280$:FILE:07,A1R
290$:ENDJOB
```

```
     OS29/N232D/SGM/RSGMTSS

REMO CLEARFILES
GET OS29/N232D/SGM/PARAMS"01",R
GET OS29/N241D/CDEP/SGMINPT2/&MANPOWERBASE."02",R
GET LA61A/SLAY/DATA/&AC-TYPE./&SPARESFILE."04",R
TEMP 07;03
RUNY OS29/N232D/SGM/CSGM,R
RUNY OS29/N232D/SGM/CPLOT,R
REMO CSGM;07;03;CPLOT;01;02;04
```

```
          OS29/N232D/SGM/SETPARAM

     REMO CLEARFILES
     TEMP 01
     GET OS29/N232D/SGM/ZD.&AC-TYPE."08",R
     RUNY OS29/N232D/SGM/HZDATA,R
     PERM 01;OS29/N232D/SGM/PARAMS
     REMO PARAMS;HZDATA;08
     OS29/N232D/SGM/ZD.F4

$ZDATA,
  INFMAN='NO',
  NONORS='NO',
  NSIM='40',
  UE='72',
  MAXFLY='72',
  RES='0',
  ATTRIT='.0',
  ANYBRK='.20',
  ANYGA='.04',
  RNMCM='.3',
  NUMDAY='30',
  NCYCLE='5',
  FTOTYM='0600',
  LTOTYM='1824',
  PREFLT='1.4',
  SRTLTH='1.7',
  SCALE='0',
  IAUGMT='NO',
$
```

APPENDIX B

SGM PROGRAM LISTING

This appendix provides a listing of the main program, subroutines, functions, and block data subprograms comprising the Sortie-Generation Model. The main program is listed first. The subroutines, function, and block data, each beginning on a new page, are then listed in alphabetical order.

```
C********** OS29/N232D/SGM/NEWSGM
C******************************************************************
C*** MAIN PROGRAM
C******************************************************************
C** MAIN       - MAIN PROGRAM FOR LMI SORTIE-GENERATION MODEL (SGM).
C***     THIS IS THE MAIN PROGRAM FOR THE LMI SORTIE GENERATION
C*** MODEL (SGM). A LIST OF ALL COMMON BLOCKS AND PARAMETER STATEMENTS
C*** USED IN THE MODEL IS PROVIDED AT THE BEGINNING OF THIS MAIN
C*** PROGRAM. THE PROCESSING SEQUENCE IS AS FOLLOWS - FIRST,
C*** ALL INPUTS ARE LOADED AND NECESSARY INITIALIZATION PERFORMED.
C*** THEN, THE ACTUAL SIMULATION IS PERFORMED, AND FINALLY THE RUN
C*** RESULTS ARE PRINTED TO THE STANDARD OUTPUT FILE.
C***
C*** INPUT FILES --
C***     01 - SCENARIO INPUT PARAMETERS
C***     02 - WORK CENTER INPUT DATA
C***     03 - SCRATCH FILE USED FOR DAILY SCENARIO PARAMETERS
C***     04 - SPARES INPUT DATA
C*** OUTPUT FILES --
C***     06 - STANDARD OUTPUT FILE (RUN RESULTS)
C***     07 - SORTIE RESULTS FOR PLOT PROGRAM
C*** PARAMETERS --
C***   MAXAC      - MAXIMUM ALLOWABLE UE-STRENGTH (# AIRCRAFT)
C***   MAXWC      - MAXIMUM ALLOWABLE NUMBER OF WORK CENTERS
C***   MAXBIT     - NUMBER OF BITS IN A COMPUTER WORD ON THIS SYSTEM
C***   MAXPRT     - MAXIMUM ALLOWABLE NUMBER OF PART-TYPES.
C***   MAXVEC     - MAXIMUM ALLOWABLE LENGTH (IN COMPUTER WORDS) OF
C***                AIRCRAFT BIT-VECTORS. A BIT-VECTOR MUST BE AT
C***                LEAST "MAXAC" BITS LONG, PLUS AN EXTRA WORD
C***                TO STORE THE AIRCRAFT COUNT FOR THAT VECTOR.
C***                HENCE, MAXVEC IS A FUNCTION OF MAXAC AND MAXBIT.
C***   MAXDAY     - MAXIMUM ALLOWABLE NUMBER OF FLYING DAYS.
C***   MAXCYC     - MAXIMUM ALLOWABLE NUMBER OF FLYING CYCLES PER DAY.
C***   MAXSTAT    - CURRENT NUMBER OF STATISTICS COLLECTED PER
C***                FLYING CYCLE PER DAY.
C***   LFLD       - LENGTH OF BIT-FIELD USED IN THE WORK-CENTER
C***                REPAIR LISTS. THIS BIT-FIELD MUST BE LARGE ENOUGH
C***                TO HOLD (MAXAC-1), THE TAIL NUMBER OF THE
C***                LAST AIRCRAFT. THUS, (2**LFLD) MUST BE GREATER
C***                THAN OR EQUAL TO MAXAC.
C***   NPERWRD    - NUMBER OF BIT-FIELDS PER COMPUTER WORD FOR THESE
C***                WORK-CENTER LISTS.THUS NPERWRD IS A FUNCTION
C***                OF LFLD AND MAXBIT.
C***   MXINWC     - LENGTH (IN COMPUTER WORDS) OF THE WORK-CENTER LISTS.
C***                MXINWC IS COMPUTED SO THAT THE MAXIMUM ALLOWABLE
C***                NUMBER OF BIT FIELDS IN A WORK-CENTER LIST IS
C***                EQUAL TO MAXAC, THE MAXIMUM NUMBER OF AIRCRAFT.
C***   IFSCEN     - FILE NUMBER OF SCENARIO INPUT FILE
C***   IFWC       - FILE NUMBER OF WORK CENTER INPUT FILE
C***   IFPRT      - FILE NUMBER OF SPARES INPUT FILE
C******************************************************************
C---
      PARAMETER  MAXAC=108,MAXWC=25,MAXBIT=36,MAXPRT=304,
&                 MAXVEC=2+(MAXAC-1)/MAXBIT
      PARAMETER LFLD=7,NPERWRD=MAXBIT/LFLD,MXINWC=1+(MAXAC-1)/NPERWRD
      PARAMETER MAXDAY=30,MAXCYC=10,MAXSTAT=5
      PARAMETER IFSCEN=01, IFWC=02, IFPRT=04
      LOGICAL INFMAN,INFPART
C---
C---/ACSTATE/  -  AIRCRAFT BIT-VECTORS.
      COMMON /ACSTATE/ LENGTH, NACVC(MAXVEC), IFLYVC(MAXVEC),
&                      MAINVC(MAXVEC), NORSVC(MAXVEC), LOSTVC(MAXVEC)
C---
C---/ALIASC/   -  TABLES FOR PART-TYPE SAMPLING.
      COMMON /ALIASC/  FRACT(MAXPRT), IALIAS(MAXPRT), FPARTS
C---
C---/BITS/     -  BIT MANIPULATION TABLES.
      COMMON /BITS/    MASK0,MASK(35), MLEFT0,MSKLFT(36),
&                      IZCOUT,ICOUNT(63)
C---
C---/DEMAND/   -  MEAN AND VARIANCE FOR TOTAL PART DEMANDS.
      COMMON /DEMAND/  ACMEAN, ACVAR, NPERAC
C---
C---/INPUT/    -  FLYING SCENARIO PARAMETERS.
      COMMON /INPUT/   INITUE, NAC, PATTRIT, IRES, RNMCM, INFPART,
&                      MAXFLY(MAXCYC), INFMAN, ISCALE, IAUGMNT
C---
C---/PARTS/    -  PART CHARACTERISTICS.
      COMMON /PARTS/   NPARTS, IQPA(MAXPRT), NBACKO(MAXPRT),
&                      BRPRATE(MAXPRT), DRPRATE(MAXPRT), INITSJ(MAXPRT),
&                      RESUPP(MAXPRT), BNRTS(MAXPRT), NBASE(MAXPRT),
&                      NDEPOT(MAXPRT)
C---
C---/RSEED/    -  SEED FOR RANDOM NUMBER GENERATOR.
      COMMON /RSEED/   SEED
C---
C---/STATS/    -  CUMULATIVE STATISTICS FOR SIMULATION RESULTS.
      COMMON /STATS/   EXPECT(MAXSTAT,MAXCYC,MAXDAY),
&                      NRESRV, IZDAY,ITOTRES(MAXDAY), LOSSTOT
C---
C---/TIME/     -  FLYING CYCLE TIMES AND SIMULATION PARAMETERS.
      COMMON /TIME/    PREFLITE, SORTLGTH, WAITCYC, TYMNITE,
&                      NSIM, ISIM, NUMDAY, IDAY, NCYCLES, ICYCLE
C---
C---/WCBRK/    -  WORK CENTER BREAK RATES.
      COMMON /WCBRK/   PACBRK, PACGABT, PBRKWC(MAXWC), PWCPROD,
&                      PBRKSEQ(2,MAXWC), INDXWC(MAXWC)
C---
C---/WCINPUT/  -  WORK CENTER INPUTS.
      COMMON /WCINPUT/ NWC, NCREWS(MAXWC), SRATE(MAXWC)
C---
C---/WCMAINT/  -  AIRCRAFT WORK CENTER LISTS.
      COMMON /WCMAINT/ LISTRP(MXINWC,MAXWC), INREPR(MAXWC)
C---
C---  *COLLECT STARTING CPU-TIME AND CORE-MEMORY REQUIREMENT
       CALL PTIME(START)
      CALL MEMSIZ(KSIZE)
C--
C--  *LOAD AND INITIALIZE SCENARIO, WORK CENTER AND PARTS DATA
      CALL INIT(IFSCEN,IFWC,IFPRT)
C--
C--  *RUN THE ACTUAL SIMULATION
      CALL SIMULA
C--
C--  *PRINT-OUT THE RESULTS OF THE SIMULATION
      CALL PRINTO
C--
C--  *PRINT MEMORY AND CPU-TIME USED
      CALL PTIME(FINISH)
      WRITE(6,9001)(FINISH-START)*60.,KSIZE
C--
   STOP
 9001 FORMAT(//,"OCPU TIME USED =",F6.2," MIN",/,
&             "OMEMORY USED   =",I6," K WORDS")
   END
```

```
C***************************************************************
   SUBROUTINE ALIAS(N,FRACT,IALIAS)
C***************************************************************
C** ALIAS      - INITIALIZE TABLES NEEDED FOR "ALIAS" SAMPLING METHOD.
C***      ALIAS IS A FORTRAN SUBROUTINE WHICH INITIALIZES THE
C*** TABLES USED BY THE ALIAS METHOD FOR GENERATING RANDOM
C*** VARIABLES FROM A DISCRETE DISTRIBUTION. SEE - "ON THE
C*** ALIAS METHOD FOR GENERATING RANDOM VARIABLES FROM A DISCRETE
C*** DISTRIBUTION" IN THE AMERICAN STATISTICIAN, NOV 1979, VOL 33,
C*** NO 4, PP 214-218, FOR A DESCRIPTION OF THIS METHOD AND THE
C*** ALGORITHM USED IN THIS ROUTINE TO CREATE THE NECESSARY TABLES.
C*** TWO TABLES ARE NEEDED FOR THIS METHOD - A TABLE OF
C*** FRACTIONAL CUTOFF VALUES AND ANOTHER FOR THE CORRESPONDING
C*** ALIASES. THE PROCEDURE USED TO GENERATE THESE TABLES IS A
C*** SINGLE-PASS, LINKED-LIST PROCEDURE.
C***
C*** INPUT -
C***   N         - NUMBER OF MASS POINTS OF THE DISCRETE DISTRIBUTION
C***               WHICH IS BEING SAMPLED.
C*** INPUT/OUTPUT -
C***   FRACT(I)  - UPON INPUT, FRACT(I) IS THE PROBABILITY
C***               DISTRIBUTION OF A RANDOM VARIABLE, R.
C***               FRACT(I)=PROBABILITY( R = I), I=1,2,...,N .
C***               UPON OUTPUT FROM THIS SUBROUTINE, FRACT CONTAINS
C***               THE TABLE OF FRACTIONAL CUTOFF VALUES USED BY THE
C***               ALIAS METHOD.
C*** OUTPUT -
C***   IALIAS(I) - TABLE OF ALIASES USED BY ALIAS METHOD. I=1,...,N
C***************************************************************
C---
      DIMENSION FRACT(N), IALIAS(N)
C---
C---        *INITIALIZE LIST HEADERS TO NO ENTRIES
             LHEAD = 0
             MHEAD = 0
C---
C---        *DO FOR(EACH POINT OF THE PROBABILITY DISTRIBUTION)
             FLOATN = FLOAT(N)
             DO 600  I=1,N
C---
C---           *INITIALIZE FRACTIONAL CUTOFF VALUE FOR THIS POINT
                FRACT(I) = FLOATN * FRACT(I)
C---
C---           *IF(THIS INDEX BELONGS IN THE "LESS" LIST, I.E. THOSE
C---               INDICES FOR WHICH FRACT(I) IS LESS THAN 1.0)THEN
                IF(FRACT(I).GE.1.0) GO TO 100
C---
C---              *ADD THIS INDEX TO HEAD OF "LESS" LIST
                   IALIAS(I) = LHEAD
                   LHEAD     = I
C---
C---           *ELSE (INDEX BELONGS IN "MORE" LIST)
                GO TO 200
```

```
 100        CONTINUE
C--
C--           *ADD INDEX TO HEAD OF "MORE" LIST
               IALIAS(I) = MHEAD
               MHEAD     = I
C--
C--         *END IF (WHICH LIST TEST)
 200        CONTINUE
C--
C--         *DO WHILE(BOTH LISTS ARE NOT EMPTY)
 300        CONTINUE
            IF(MHEAD.EQ.0) GO TO 500
            IF(LHEAD.EQ.0) GO TO 500
C--
C--           *REMOVE NEXT INDEX FROM "LESS" LIST
               LNEXT = LHEAD
               LHEAD = IALIAS(LHEAD)
C--
C--           *SET ALIAS FOR THIS INDEX TO NEXT ENTRY IN "MORE" LIST
               IALIAS(LNEXT) = MHEAD
C--
C--           *UPDATE CUTOFF VALUE FOR THIS "MORE" ENTRY
               FRACT(MHEAD) = FRACT(MHEAD) - (1.0-FRACT(LNEXT))
C--
C--           *IF(THIS INDEX NO LONGER BELONGS IN "MORE" LIST)THEN
               IF(FRACT(MHEAD).GE. 1.0) GO TO 400
C--
C--             *REMOVE INDEX FROM "MORE" LIST AND ADD IT TO "LESS"
                 LNEXT         = MHEAD
                 MHEAD         = IALIAS(MHEAD)
                 IALIAS(LNEXT) = LHEAD
                 LHEAD         = LNEXT
C--
C--           *END IF (SWITCH LISTS TEST)
 400           CONTINUE
C--
C--         *END DO (LISTS LOOP)
            GO TO 300
 500        CONTINUE
C--
C--       *END DO (INDEX LOOP)
 600       CONTINUE
C--
C--       *ADJUST FRACT(I) TO SAVE TIME IN MNOM SUBROUTINE.
           DO 700 I=1,N
            FRACT(I) = FRACT(I) + (I-1)
 700       CONTINUE
C--
  RETURN
  END
```

```
C***************************************************************
   SUBROUTINE ATTRIT(PLOST,NTOFLY,IFLYVC,LOSTVC,NLOST)
C***************************************************************
C++ ATTRIT     - SIMULATE ATTRITION PROCESS DURING A SORTIE PERIOD.
C***     ATTRIT IS A FORTRAN SUBROUTINE WHICH SIMULATES THE
C*** EFFECTS OF ATTRITION DURING A SORTIE. ATTRIT DRAWS A RANDOM
C*** SAMPLE FROM A BINOMIAL DISTRIBUTION BASED ON THE NUMBER OF
C*** AIRCRAFT FLYING THE SORTIE AND THE PROBABILITY OF ATTRITION
C*** GIVEN AN AIRCRAFT FLIES A SORTIE. ATTRIT THEN SELECTS THE
C*** RIGHTMOST AIRCRAFT FROM THE CURRENT FLYABLE AIRCRAFT VECTOR AS
C*** THE AIRCRAFT WHICH WERE ATTRITED. THE RIGHTMOST ONES ARE SELECTED
C*** TO SPEED UP COMPUTATION IN OTHER ROUTINES.
C***
C*** INPUT -
C***   PLOST      - PROBABILITY THAT AN AIRCRAFT ATTRITS GIVEN THAT IT
C***                FLIES A SORTIE.
C*** INPUT/OUTPUT -
C***   NTOFLY     - NO. OF A/C TO FLY THIS PERIOD.
C***   IFLYVC     - FLYABLE AIRCRAFT STATUS VECTOR. INDICATES THOSE
C***                AIRCRAFT WHICH ARE STILL FLYABLE DURING THE CURRENT
C***                FLYING CYCLE, I.E. THOSE AIRCRAFT WHICH WERE FLYABLE
C***                AT THE START OF PREFLIGHT AND HAVE NOT GROUND-ABORTED,
C***                ATTRITED, OR BROKEN THUS FAR IN THE CYCLE.
C***                THE FIRST WORD, IFLYVC(1), CONTAINS THE TOTAL
C***                NUMBER OF AIRCRAFT STILL FLYABLE THUS FAR IN
C***                THE CURRENT FLYING CYCLE. THE REMAINDER OF THE
C***                ARRAY IS A BIT VECTOR WITH EACH BIT REPRESENTING
C***                AN AIRCRAFT. A 1-BIT INDICATES THE AIRCRAFT IS
C***                STILL FLYABLE. NOTE THAT IFLYVC(1) ALSO INDICATES
C***                THE NUMBER OF 1-BITS IN THIS BIT VECTOR.
C***   LOSTVC     - ATTRITED AIRCRAFT VECTOR. INDICATES THOSE AIR-
C***                CRAFT WHICH HAVE ATTRITED THUS FAR IN THE SIMULATION
C***                AND NOT BEEN REPLACED BY RESERVES. THE FIRST WORD,
C***                LOSTVC(1), CONTAINS THE TOTAL NUMBER OF AIRCRAFT
C***                WHICH HAVE BEEN LOST AND NOT REPLACED BY RESERVES.
C***                THE REMAINDER OF THE ARRAY IS A BIT VECTOR WITH
C***                EACH BIT REPRESENTING AN AIRCRAFT. A 1-BIT INDICATES
C***                THE AIRCRAFT HAS BEEN ATTRITTED. NOTE THAT
C***                LOSTVC(1) ALSO INDICATES THE NUMBER OF 1-BITS IN
C***                THIS BIT VECTOR.
C*** OUTPUT -
C***   NLOST      - NUMBER OF AIRCRAFT LOST ON THIS SORTIE
C***************************************************************
C---
C---      *DETERMINE NUMBER OF ATTRITIONS BY SAMPLING FROM THE
C---                APPROPRIATE BINOMIAL DISTRIBUTION
           NLOST = NBINOM(PLOST,NTOFLY)
C---
C---      *IF(ANY AIRCRAFT WERE ATTRITED)THEN
           IF(NLOST .EQ. 0)  GO TO 1000
C---
C---         *REDUCE NO. OF A/C CAPABLE OF FLYING THIS PERIOD
              NTOFLY = NTOFLY - NLOST
```

```
C--
C--           *TRANSFER RIGHTMOST AIRCRAFT FROM FLYABLE AIRCRAFT
C--                VECTOR TO THE ATTRITED AIRCRAFT VECTOR
              CALL TBITSR(NLOST,IFLYVC,LOSTVC)
C--
C--        *END IF (ZERO ATTRITIONS TEST)
 1000       CONTINUE
C--
   RETURN
   END
```

```
C*****************************************************************
   BLOCK DATA
C*****************************************************************
C** BLOCK DATA - INITIALIZES COMMON TABLES FOR BIT MANIPULATIONS.
C***     THIS SUBPROGRAM INITIALIZES THE TABLES CONTAINED IN
C*** THE /BIT/ COMMON BLOCK.  THIS INITIALIZATION IS DONE
C*** DURING COMPILATION; THE SUBPROGRAM CONTAINS NO
C*** EXECUTABLE STATEMENTS. THE /BIT/ COMMON BLOCK CONTAINS
C*** THREE SETS OF TABLES WHICH ARE USED FOR ACCESSING BITS AND
C*** BIT FIELDS WITHIN A COMPUTER WORD. NOTE THAT THE FOLLOWING
C*** PROGRAMMING TECHNIQUE IS USED IN EACH OF THESE TABLES - AN
C*** EXTRA WORD IS PLACED BEFORE THE BEGINNING OF EACH TABLE. THIS
C*** EXTRA WORD REPRESENTS TABLE(0), I.E., THE 0TH INDEXED WORD IN
C*** THE TABLE. THUS , THE TABLE IS ACTUALLY INDEXED 0,1,2,...
C*** THIS TECHNIQUE OF REFERENCING THE 0TH WORD OF AN ARRAY IS
C*** NOT STANDARD FORTRAN AND MAY NOT WORK WITH OTHER FORTRAN COMPILERS.
C*** THESE TABLES REMAIN FIXED THROUGHOUT THE SIMULATION.
C***
C*** COMMON TABLES --
C***    MASK(I)     - I=0,1,...,35.  MASK IS THE BIT ACCESSING TABLE
C***                  USED IN THE SGM.  THE BITS IN THE COMPUTER WORD
C***                  ARE NUMBERED, LEFT TO RIGHT, 0,1,2,...,35,
C***                  AND MASK(I) HAS A 1-BIT IN THE ITH POSITION AND
C***                  ZEROES ELSEWHERE. THIS TABLE IS USED TO MASK-OFF
C***                  THE ITH BIT IN A COMPUTER WORD.
C***    MSKLFT(I)   - I=0,1,...,36 . MSKLFT IS USED TO MASK-OFF THE
C***                  LEFTMOST BITS IN A COMPUTER WORD. THE FIRST
C***                  (LEFTMOST) I BITS OF MSKLFT(I) ARE 1-BITS
C***                  AND THE REMAINING BITS ARE ZERO. THUS, FOR
C***                  EXAMPLE, MSKLFT(0) WOULD BE ALL 0S AND
C***                  MSKLFT(36) WOULD BE ALL 1S.
C***    ICOUNT(I)   - I=0,1,...,63.  THIS IS A TABLE WHICH IS USED TO
C***                  COUNT THE NUMBER OF 1-BITS IN ANY GIVEN 6-BIT
C***                  FIELD. IN A 6-BIT FIELD, THERE ARE 2**6 = 64
C***                  POSSIBLE BIT PATTERNS -- THE BINARY
C***                  REPRESENTATIONS OF THE INTEGERS 0,1,2,...63.
C***                  ICOUNT(I) CONTAINS THE NUMBER OF 1-BITS IN THE
C***                  BINARY REPRESENTATION OF I, E.G., ICOUNT(3)=2 .
C***                  THIS TABLE IS USEFUL IN COUNTING THE NUMBER OF
C***                  1-BITS REPRESENTING AIRCRAFT IN THE VARIOUS
C***                  AIRCRAFT-STATUS BIT-VECTORS. THIS TECHNIQUE
C***                  IS MUCH FASTER THAN A BIT-BY-BIT COUNT.
C*****************************************************************
C---
      COMMON /BITS/ MASK0,MASK(35), MLEFT0,MSKLFT(36),
&                                IZCOUT,ICOUNT(63)
C---
      DATA MASK0/0400000000000 /
      DATA MASK/                 0200000000000, 0100000000000,
&               040000000000 , 020000000000 , 010000000000 ,
&               04000000000  , 02000000000  , 01000000000  ,
&               0400000000   , 0200000000   , 0100000000   ,
&               040000000    , 020000000    , 010000000    ,
&              04000000     , 02000000     , 01000000     ,
&              0400000      , 0200000      , 0100000      ,
&              040000       , 020000       , 010000       ,
&              04000        , 02000        , 01000        ,
&              0400         , 0200         , 0100         ,
&              040          , 020          , 010          ,
&              04           , 02           , 01           /
C---
      DATA MLEFT0/0/
      DATA MSKLFT/040000000000, 060000000000, 070000000000,
&                 074000000000, 076000000000, 077000000000,
&                 077400000000, 077600000000, 077700000000,
&                 077740000000, 077760000000, 077770000000,
&                 077774000000, 077776000000, 077777000000,
&                 077777400000, 077777600000, 077777700000,
&                 077777740000, 077777760000, 077777770000,
&                 077777774000, 077777776000, 077777777000,
&                 077777777400, 077777777600, 077777777700,
&                 077777777740, 077777777760, 077777777770,
&                 077777777774, 077777777776, 077777777777 /
C---
      DATA IZCOUT/0/
      DATA ICOUNT/     1,  1,  2,  1,  2,  2,  3,  1,  2,  2,
&                      3,  2,  3,  3,  4,  1,  2,  2,  3,  2,
&                      3,  3,  4,  2,  3,  3,  4,  3,  4,  4,
&                      5,  1,  2,  2,  3,  2,  3,  3,  4,  2,
&                      3,  3,  4,  3,  4,  4,  5,  2,  3,  3,
&                      4,  3,  4,  4,  5,  3,  4,  4,  5,  4,
&                      5,  5,  6  /
C---
    END
```

```
C*************************************************************************
   SUBROUTINE BREAK(PBREAK,PBRKSEQ,INDXWC,NTOFLY,IFLYVC,NORSVC)
C*************************************************************************
C** BREAK      - SIMULATE AIRCRAFT BREAKS AFTER A SORTIE.
C***    BREAK IS A FORTRAN SUBROUTINE WHICH SIMULATES THE PROCESS
C*** OF AIRCRAFT BREAKING UPON RETURNING FROM A SORTIE. GIVEN A NUMBER
C*** OF FLYABLE AIRCRAFT AND THE OVERALL BREAK RATE, THIS ROUTINE FIRST
C*** DETERMINES THE NUMBER OF AIRCRAFT WHICH BROKE. IT THEN DETERMINES
C*** THE NUMBER AND DISTRIBUTION OF PARTS DEMANDS RESULTING FROM THESE
C*** BREAKS. THEN ASSUMING IMMEDIATE AND MAXIMUM CANNABILIZATION, THE
C*** NUMBER OF NORS AIRCRAFT AMONG THESE BROKEN AIRCRAFT IS DETERMINED.
C*** THOSE AIRCRAFT WHICH ARE NOT NORS ARE PROBABILISTICALLY BROKEN
C*** DIRECTLY INTO THE VARIOUS WORKCENTERS.
C***
C*** INPUT -
C***   PBREAK     - PROBABILITY THAT A FLYABLE AIRCRAFT BREAKS
C***                INTO AT LEAST ONE WORKCENTER UPON RETURNING
C***                FROM A SORTIE.
C***   PBRKSEQ    - 2-DIMENSIONAL ARRAY USED TO DETERMINE THE
C***                DISTRIBUTION OF ABORTS INTO THE VARIOUS
C***                WORKCENTERS.
C***   INDXWC     - AN INDEX ARRAY USED TO DETERMINE THE DISTRIBUTION
C***                OF BREAKS INTO THE VARIOUS WORK CENTERS.
C*** INPUT/OUTPUT -
C***   NTOFLY     - NO. OF A/C TO FLY THIS PERIOD.
C***   IFLYVC     - FLYABLE AIRCRAFT STATUS VECTOR. INDICATES THOSE
C***                AIRCRAFT WHICH ARE STILL FLYABLE DURING THE CURRENT
C***                FLYING CYCLE. I.E. THOSE AIRCRAFT WHICH WERE FLYABLE
C***                AT THE START OF PREFLIGHT AND HAVE NOT GROUND-ABORTED,
C***                ATTRITED, OR BROKEN THUS FAR IN THE CYCLE.
C***                THE FIRST WORD, IFLYVC(1), CONTAINS THE TOTAL
C***                NUMBER OF AIRCRAFT STILL FLYABLE THUS FAR IN
C***                THE CURRENT FLYING CYCLE. THE REMAINDER OF THE
C***                ARRAY IS A BIT VECTOR WITH EACH BIT REPRESENTING
C***                AN AIRCRAFT. A 1-BIT INDICATES THE AIRCRAFT IS
C***                STILL FLYABLE. NOTE THAT IFLYVC(1) ALSO INDICATES
C***                THE NUMBER OF 1-BITS IN THIS BIT VECTOR.
C***   NORSVC     - NORS AIRCRAFT STATUS VECTOR. INDICATES THOSE
C***                AIRCRAFT WHICH ARE NORS DUE TO UNAVAILABLE PARTS.
C***                THE FIRST WORD, NORSVC(1), CONTAINS THE TOTAL
C***                NUMBER OF 1-BITS IN THE NORS STATUS VECTOR.
C***                THE REMAINDER OF THE ARRAY IS A BIT STRING WITH
C***                EACH BIT REPRESENTING AN AIRCRAFT. A 1 INDICATES
C***                THE AIRCRAFT IS NORS. NOTE THAT NORSVC(1) ALSO
C***                INDICATES THE NUMBER OF 1-BITS IN THIS BIT STRING.
C*************************************************************************
C---
     DIMENSION NORSVC(1)
C---
C---       *IF(THERE ARE STILL ANY FLYABLE AIRCRAFT)THEN
           IF(NTOFLY.EQ.0) GO TO 4000
C---
C---          *DETERMINE NUMBER OF AIRCRAFT BREAKING INTO WORKCENTERS
```

```
C---             BY SAMPLING FROM THE APPROPRIATE BINOMIAL DISTRIBUTION
             NTOTBK = NBINOM(PBREAK,NTOFLY)
C---
C---         *IF(THERE ARE ANY BROKEN AIRCRAFT)THEN
             IF(NTOTBK.EQ.0) GO TO 3000
C---
C---           *REDUCE NO. OF A/C CAPABLE OF FLYING THIS PERIOD
                NTOFLY = NTOFLY - NTOTBK
C---
C---           *DETERMINE NUMBER/DISTRIBUTION OF PARTS DEMANDS RESULTING
C---            FROM THESE BROKEN AIRCRAFT AND DETERMINE NEW NUMBER OF
C---            NORS AIRCRAFT AFTER IMMEDIATE AND MAXIMUM CANNABILIZATION
                NEWNOR = NORSBK(NTOTBK,NORSVC(1))
                NORDIF = NEWNOR - NORSVC(1)
C---
C---           *IF(NOT ALL THE BROKEN AIRCRAFT ARE NORS)THEN
                IF(NORDIF.GE.NTOTBK) GO TO 1000
C---
C---              *BREAK THE LEFTMOST FLYABLE AIRCRAFT INTO MAINTENANCE
                   CALL WCDIST(NTOTBK-NORDIF,PBRKSEQ,INDXWC,IFLYVC)
C---
C---           *END IF (ALL NORS TEST)
 1000           CONTINUE
C---
C---           *IF(SOME OF THE BROKEN AIRCRAFT ARE NORS)THEN
                IF(NORDIF.LE.0) GO TO 2000
C---
C---              *TRANSFER LEFTMOST AIRCRAFT FROM FLYABLE STATUS
C---                          VECTOR TO THE NORS STATUS VECTOR
                   CALL TBITSL(NORDIF,IFLYVC,NORSVC)
C---
C---           *END IF (NONZERO NORS TEST)
 2000           CONTINUE
C---
C---         *END IF (ZERO BREAKS TEST)
 3000        CONTINUE
C---
C---      *END IF (ZERO FLYABLE AIRCRAFT TEST)
 4000     CONTINUE
C---
   RETURN
   END
```

```
C*************************************************************************
      SUBROUTINE CRESERV(IAUGMNT,LOSTVC,NRESRV,NAC)
C*************************************************************************
C++ CRESERV    - COMMIT RESERVE AIRCRAFT.
C***    CRESERV IS A FORTRAN SUBROUTINE WHICH WILL ALLOCATE
C*** RESERVE AIRCRAFT TO REPLACE THOSE AIRCRAFT WHICH HAVE BEEN
C*** LOST DUE TO ATTRITION. IF ENOUGH RESERVES ARE LEFT, ALL LOSSES
C*** ARE REPLACED; HENCE THE ATTRITION VECTOR IS ZEROED OUT. IF
C*** THERE ARE NOT ENOUGH RESERVES TO COVER ALL THE LOSSES, THEN
C*** THE ATTRITIONS ON THE LEFT OF THE ATTRITION VECTOR ARE
C*** REPLACED. THIS ARBITRARY SELECTION WILL HELP TO SPEED UP THE
C*** SELECTION ROUTINES. NOTE THAT ALL RESERVES ARE ASSUMED TO
C*** BE FULLY MISSION CAPABLE (I.E. FLYABLE) WHEN COMMITTED.
C***
C*** INPUT --
C***   IAUGMNT    - FLAG INDICATING WHETHER RESERVES ARE TO BE USED
C***                ONLY AS ATTRITION FILLERS OR TO AUGMENT THE
C***                CURRENT UE-STRENGTH. IF IAUGMNT=0, RESERVES
C***                ARE USED ONLY TO REPLACE COMBAT LOSSES; HENCE
C***                NOT ALL RESERVES MAY BE COMMITTED WHEN THEY
C***                BECOME AVAILABLE. IF IAUGMNT=1, ALL RESERVES
C***                ARE COMMITTED IMMEDIATELY UPON BECOMING
C***                AVAILABLE. THIS FLAG IS A USER-SPECIFIED INPUT
C***                WHICH REMAINS FIXED THROUGHOUT THE SIMULATION.
C*** INPUT/OUTPUT --
C***   LOSTVC     - ATTRITED AIRCRAFT VECTOR. INDICATES THOSE AIR-
C***                CRAFT WHICH HAVE ATTRITED THUS FAR IN THE SIMULATION
C***                AND NOT BEEN REPLACED BY RESERVES. THE FIRST WORD,
C***                LOSTVC(1), CONTAINS THE TOTAL NUMBER OF AIRCRAFT
C***                WHICH HAVE BEEN LOST AND NOT REPLACED BY RESERVES.
C***                THE REMAINDER OF THE ARRAY IS A BIT VECTOR WITH
C***                EACH BIT REPRESENTING AN AIRCRAFT. A 1-BIT INDICATES
C***                THE AIRCRAFT HAS BEEN ATTRITTED. NOTE THAT
C***                LOSTVC(1) ALSO INDICATES THE NUMBER OF 1-BITS IN
C***                THIS BIT VECTOR.
C***   NRESRV     - NUMBER OF AIRCRAFT CURRENTLY IN RESERVE.
C***   NAC        - CURRENT UE-STRENGTH. IF THE AUGMENT FLAG
C***                IS SET, THE RESERVES WHICH REMAIN AFTER
C***                REPLACING COMBAT LOSSES WILL BE USED TO
C***                AUGMENT THIS CURRENT UE-STRENGTH. IF THE
C***                FLAG IS NOT SET OR THERE ARE NOT ENOUGH
C***                RESERVES TO COVER ALL THE COMBAT LOSSES, THEN
C***                NAC WILL REMAIN UNCHANGED.
C*************************************************************************
C---
      DIMENSION LOSTVC(1)
C---
C--- *REPLACE AS MANY AIRCRAFT LOSSES AS POSSIBLE
      NFILLS = MIN0(LOSTVC(1),NRESRV)
      CALL ZBITSL(NFILLS,LOSTVC)
      NRESRV = NRESRV - NFILLS
C---
C--- *IF(THERE ARE STILL REMAINING RESERVES AND EXCESS RESERVES
C---                               ARE TO BE AUGMENTED)THEN
      IF(NRESRV.LE.0)GO TO 100
      IF(IAUGMNT.EQ.0)GO TO 100
C---
C---     *INCREASE UE BY AUGMENTING REMAINING RESERVES
          NAC = NAC + NRESRV
          CALL UEUPDAT(NAC)
C---
C---     *SET REMAINING RESERVES TO NONE
          NRESRV = 0
C---
C--- *END IF (AUGMENTATION TEST)
  100 CONTINUE
C---
      RETURN
      END
```

```
C*****************************************************************
      SUBROUTINE FLYCYC
C*****************************************************************
C** FLYCYC    - SIMULATE AIRCRAFT FLYING CYCLE.
C***     THIS ROUTINE IS THE BASIC LOGICAL STRUCTURE OF THE SGM
C*** SIMULATION. THE VARIOUS DISCRETE EVENTS WHICH CAN OCCUR,
C*** GROUND-ABORTS, AIRCRAFT REPAIRS, BREAKS, ETC., ARE
C*** STRUCTURED ACCORDING TO A USER-SPECIFIED FLYING CYCLE
C*** CONSISTING OF A MINIMAL-RECOVERY PERIOD, A SORTIE-PERIOD,
C*** AND EITHER A WAIT OR AN OVERNIGHT PERIOD. A SPECIFIED SEQUENCE
C*** OF THESE FLYING CYCLES COMPRISE A FLYING DAY, AND THE
C*** SIMULATION CONSISTS OF A SEQUENCE OF FLYING DAYS.
C***     THE NUMEROUS INPUTS AND OUTPUTS OF THIS ROUTINE ARE
C*** ALL CONTAINED IN COMMON BLOCKS. DEFINITIONS ARE PROVIDED
C*** IN THE VARIOUS ROUTINES CALLED BY FLYCYC AND WILL NOT
C*** BE REPEATED HERE.
C*****************************************************************
C---
      PARAMETER  MAXAC=108,MAXWC=25,MAXBIT=36,
     &             MAXVEC=2+(MAXAC-1)/MAXBIT
      PARAMETER MAXDAY=30,MAXCYC=10,MAXSTAT=5
      LOGICAL INFMAN,INFPART
      COMMON /ACSTATE/ LENGTH, NACVC(MAXVEC), IFLYVC(MAXVEC),
     &                 MAINVC(MAXVEC), NORSVC(MAXVEC), LOSTVC(MAXVEC)
      COMMON /INPUT/   INITUE, NAC, PATTRIT, IRES, RNMCM, INFPART,
     &                 MAXFLY(MAXCYC), INFMAN, ISCALE, IAUGMNT
      COMMON /STATS/   EXPECT(MAXSTAT,MAXCYC,MAXDAY),
     &                 NRESRV, IZDAY,ITOTRES(MAXDAY), LOSSTOT
      COMMON /TIME/    PREFLITE, SORTLGTH, WAITCYC, TYMNITE,
     &                 NSIM, ISIM, NUMDAY, IDAY, NCYCLES, ICYCLE
      COMMON /WCBRK/   PACBRK, PACGABT, PBRKWC(MAXWC), PWCPROD,
     &                 PBRKSEQ(2,MAXWC), INDXWC(MAXWC)
      COMMON /WCINPUT/ NWC, NCREWS(MAXWC), SRATE(MAXWC)
C---
C---  *UPDATE OVERALL MAINTENANCE BIT-VECTOR FOR THIS FLYING CYCLE
       CALL MUPDATE(NWC,MAINVC)
C---
C---  *COMPUTE NUMBER OF FULLY-MISSION-CAPABLE AIRCRAFT AND NUMBER
C---            OF AIRCRAFT TO SCHEDULE FOR NEXT SORTIE
C---   (A MISSION-CAPABLE AIRCRAFT IS DEFINED AS ONE NOT IN
C---    MAINTENANCE, NORS, OR COMBAT LOSS STATES)
       DO 100 L=2,LENGTH
          IFLYVC(L) = XOR (NACVC(L),LOSTVC(L),MAINVC(L),NORSVC(L))
  100  CONTINUE
       IFLYVC(1) = N1VECT (IFLYVC)
       NTOFLY = MINO (MAXFLY(ICYCLE),IFLYVC(1))
C---
C---  *AIRCRAFT-REPAIR EVENT -- DETERMINE AIRCRAFT REPAIRED IN
C---       EACH WORK-CENTER DURING MINIMAL-RECOVERY PERIOD
       CALL REPAIR(PREFLITE,NWC,NCREWS,SRATE)
C---
C---  *GROUND-ABORT EVENT -- DETERMINE NUMBER OF GROUND ABORTS
       CALL GABORT(PACGABT,PBRKSEQ,INDXWC,NTOFLY,IFLYVC)
```

```
C--
C--  *STATISTICS EVENT -- UPDATE CUMULATIVE STATISTICS
C--       (STATISTICS ARE ALWAYS COLLECTED AT THE BEGINNING
C--       OF THE SORTIE PERIOD.  THE MAINTENANCE STAT
C--       REFLECTS ONLY THOSE AIRCRAFT IN MAINTENANCE AT THE
C--       START OF THE MINIMAL-RECOVERY PERIOD AND DOES
C--       NOT ACCOUNT FOR GROUND-ABORTS OR REPAIRS DURING
C--       THIS MINIMAL RECOVERY PERIOD)
      EXPECT(1,ICYCLE,IDAY) = EXPECT(1,ICYCLE,IDAY) + NTOFLY
      EXPECT(2,ICYCLE,IDAY) = EXPECT(2,ICYCLE,IDAY) + MAINVC(1)
      EXPECT(3,ICYCLE,IDAY) = EXPECT(3,ICYCLE,IDAY) + NORSVC(1)
      EXPECT(4,ICYCLE,IDAY) = EXPECT(4,ICYCLE,IDAY) + LOSSTOT
      EXPECT(5,ICYCLE,IDAY) = EXPECT(5,ICYCLE,IDAY) + NRESRV
C--
C--  *IF(THIS IS NOT THE LAST CYCLE OF THE LAST DAY)THEN
C--           (NONE OF THE FOLLOWING WORK WILL AFFECT
C--            THE OUTPUT RESULTS IF THIS IS THE LAST SORTIE)
      IF((ICYCLE.GE.NCYCLES).AND.(IDAY.GE.NUMDAY))GO TO 400
C--
C--     *ATTRITION EVENT -- DETERMINE NUMBER OF ATTRITED AIRCRAFT
         CALL ATTRIT(PATTRIT,NTOFLY,IFLYVC,LOSTVC,NLOST)
         LOSSTOT = LOSSTOT + NLOST
C--
C--     *REPAIR EVENT -- DETERMINE NUMBER OF AIRCRAFT REPAIRED IN EACH
C--          WORK CENTER DURING THE SORTIE PERIOD
         CALL REPAIR(SORTLGTH,NWC,NCREWS,SRATE)
C--
C--     *IF(IT IS NOT THE LAST SORTIE OF THE FLYING DAY)THEN
         IF(ICYCLE.EQ.NCYCLES) GO TO 200
C--
C--        *BREAK-EVENT -- DETERMINE AIRCRAFT BREAKS AND PART DEMANDS
            CALL BREAK(PACBRK,PBRKSEQ,INDXWC,NTOFLY,IFLYVC,NORSVC)
C--
C--        *AIRCRAFT-REPAIR EVENT --DETERMINE AIRCRAFT REPAIRED IN
C--             EACH WORK-CENTER DURING THE WAIT PERIOD
            CALL REPAIR(WAITCYC,NWC,NCREWS,SRATE)
C--
C--     *ELSE (THIS IS THE OVERNIGHT PERIOD)
         GO TO 300
  200    CONTINUE
C--
C--        *PARTS-REPAIR EVENT -- DETERMINE SPARE PARTS REPAIRED
C--                 IN THE LAST 24-HOUR PERIOD
            CALL PRTREP(24.,PBRKSEQ,INDXWC,NORSVC)
C--
C--        *BREAK-EVENT -- DETERMINE AIRCRAFT BREAKS AND PART DEMANDS
            CALL BREAK(PACBRK,PBRKSEQ,INDXWC,NTOFLY,IFLYVC,NORSVC)
C--
C--        *AIRCRAFT-REPAIR EVENT - DETERMINE OVERNIGHT AIRCRAFT REPAIRS
            CALL REPAIR(TYMNITE,NWC,NCREWS,SRATE)
C--
C--        *COMMIT-RESERVES EVENT -- BRING-IN AVAILABLE FULLY-MISSION-
C--             CAPABLE RESERVE AIRCRAFT TO REPLACE ANY COMBAT LOSSES
```

```
            CALL CRESERV(IAUGMNT,LOSTVC,NRESRV,NAC)
C---
C---      *END IF (OVERNIGHT PERIOD TEST)
  300       CONTINUE
C---
C---   *END IF(LAST-CYCLE-OF-LAST-DAY TEST)
  400   CONTINUE
C---
    RETURN
    END
```

```
C*************************************************************
   SUBROUTINE GABORT(PABORT,PBRKSEQ,INDXWC,NTOFLY,IFLYVC)
C*************************************************************
C** GABORT     - SIMULATE AIRCRAFT GROUND-ABORT PROCESS.
C***    GABORT IS A FORTRAN SUBROUTINE WHICH SIMULATES THE PROCESS
C*** OF AIRCRAFT GROUND-ABORTING INTO WORKCENTERS AT THE END OF PREFLIGHT.
C*** GIVEN A NUMBER OF FLYABLE AIRCRAFT AND THE OVERALL GROUND-ABORT
C*** RATE, THIS ROUTINE CALCULATES THE TOTAL NUMBER OF GROUND-ABORTS,
C*** AND THEN DETERMINES WHICH WORKCENTERS THESE AIRCRAFT BROKE INTO.
C***
C*** INPUT -
C***   PABORT     - PROBABILITY THAT A FLYABLE AIRCRAFT GROUND-ABORTS
C***                INTO AT LEAST ONE WORKCENTER DURING PREFLIGHT.
C***   PBRKSEQ    - 2-DIMENSIONAL ARRAY USED TO DETERMINE THE
C***                DISTRIBUTION OF ABORTS INTO THE VARIOUS
C***                WORKCENTERS.
C***   INDXWC     - AN INDEX ARRAY USED TO DETERMINE THE DISTRIBUTION
C***                OF BREAKS INTO THE VARIOUS WORK CENTERS.
C*** INPUT/OUTPUT -
C***   NTOFLY     - NO. OF A/C TO FLY THIS PERIOD.
C***   IFLYVC     - FLYABLE AIRCRAFT STATUS VECTOR. INDICATES THOSE
C***                AIRCRAFT WHICH ARE STILL FLYABLE DURING THE CURRENT
C***                FLYING CYCLE. I.E. THOSE AIRCRAFT WHICH WERE FLYABLE
C***                AT THE START OF PREFLIGHT AND HAVE NOT GROUND-ABORTED,
C***                ATTRITED, OR BROKEN THUS FAR IN THE CYCLE.
C***                THE FIRST WORD, IFLYVC(1), CONTAINS THE TOTAL
C***                NUMBER OF AIRCRAFT STILL FLYABLE THUS FAR IN
C***                THE CURRENT FLYING CYCLE. THE REMAINDER OF THE
C***                ARRAY IS A BIT VECTOR WITH EACH BIT REPRESENTING
C***                AN AIRCRAFT. A 1-BIT INDICATES THE AIRCRAFT IS
C***                STILL FLYABLE. NOTE THAT IFLYVC(1) ALSO INDICATES
C***                THE NUMBER OF 1-BITS IN THIS BIT VECTOR.
C*************************************************************
C---
C---      *DETERMINE NUMBER OF AIRCRAFT GROUND-ABORTING INTO WORKCENTERS
C---           BY SAMPLING FROM THE APPROPRIATE BINOMIAL DISTRIBUTION
           NTOTBK = NBINOM(PABORT,NTOFLY)
C---
C---      *IF(THERE ARE ANY BROKEN AIRCRAFT)THEN
           IF(NTOTBK.EQ.0) GO TO 1000
C---
C---         *REDUCE NO. OF A/C CAPABLE OF FLYING THIS PERIOD
              NTOFLY = NTOFLY - NTOTBK
C---
C---         *BREAK THE LEFTMOST FLYABLE AIRCRAFT INTO MAINTENANCE
              CALL WCDIST(NTOTBK,PBRKSEQ,INDXWC,IFLYVC)
C---
C---      *END IF (ZERO BREAKS TEST)
 1000      CONTINUE
C---
   RETURN
   END
```

```
C**************************************************************
   SUBROUTINE INIT(IFSCEN,IFWC,IFPRT)
C**************************************************************
C** INIT       - INITIALIZE SGM SIMULATION.
C***    THIS ROUTINE READS AND INITIALIZES THE VARIABLES FOR AN
C*** SGM RUN.  IT PERFORMS THE FOLLOWING SERIES OF STEPS -
C*** 1) LOAD AND SET THE VARIOUS PARAMETERS DESCRIBING THE RUN SCENARIO,
C*** 2) LOAD AIRCRAFT MAINTENANCE MANPOWER INPUTS, 3) LOAD
C*** SPARE PARTS INFORMATION, AND 4) SET MISCELLANEOUS PARAMETERS
C*** FOR MODEL USE.  EACH INPUT FILE IS CLOSED IMMEDIATELY AFTER
C*** ALL OF ITS INFORMATION HAS BEEN READ.
C***
C*** INPUTS --
C***   IFSCEN      - INPUT FILE CONTAINING SCENARIO PARAMETERS
C***   IFWC        - INPUT FILE CONTAINING MAINTENANCE MANPOWER INPUTS
C***   IFPRT       - INPUT FILE CONTAINING SPARE PARTS DATA
C*** COMMON INPUTS --
C***   INFPART     - LOGICAL FLAG INDICATING WHETHER INFINITE PARTS
C***                 ASSUMPTION HOLDS.
C***   INFMAN      - LOGICAL FLAG INDICATING WHETHER INFINITE MANPOWER
C***                 ASSUMPTION IS BEING MADE.
C***   SORTLGTH    - LENGTH (IN HOURS) OF EACH SORTIE.
C***   PACBRK      - AIRCRAFT BREAK RATE.
C***   NAC         - CURRENT UE STRENGTH.
C*** COMMON OUTPUT --
C***   EXPECT(I,J,K) - CUMULATIVE STATISTICS ARRAY.
C**************************************************************
C--
      PARAMETER MAXCYC=10, MAXSTAT=5, MAXDAY=30, MAXWC=25
      COMMON /INPUT/   INITUE, NAC, PATTRIT, IRES, RNMCM, INFPART,
&                      MAXFLY(MAXCYC), INFMAN, ISCALE, IAUGMNT
      COMMON /STATS/   EXPECT(MAXSTAT,MAXCYC,MAXDAY),
&                      NRESRV, IZDAY,ITOTRES(MAXDAY), LOSSTOT
      COMMON /TIME/    PREFLITE, SORTLGTH, WAITCYC, TYMNITE,
&                      NSIM, ISIM, NUMDAY, IDAY, NCYCLES, ICYCLE
      COMMON /WCBRK/   PACBRK, PACGABT, PBRKWC(MAXWC), PWCPROD,
&                      PBRKSEQ(2,MAXWC), INDXWC(MAXWC)
C--
C--   *LOAD AND SET SCENARIO INPUT PARAMETERS
       CALL INITSCN(IFSCEN)
C--
C--   *LOAD AND INITIALIZE SPARE-PARTS DATA
       CALL INITPRT(IFPRT,INFPART,SORTLGTH,PACBRK)
C--
C--   *READ AIRCRAFT MAINTENANCE MANPOWER INPUTS
       CALL INITWC(IFWC,INFMAN,NAC)
C--
C--   *ZERO-OUT CUMULATIVE-STATISTICS ARRAY
       CALL ZERO(EXPECT,MAXSTAT*MAXCYC*MAXDAY)
C--
   RETURN
   END
```

```
C*************************************************************
      SUBROUTINE INITBO(NAC)
C*************************************************************
C++ INITBO     - INITIALIZE PARTS IN RESUPPLY AT START OF SIMULATION.
C***     INITBO INITIALIZES THE NUMBER OF BACKORDERS
C*** FOR EACH PART TYPE AT THE START OF EACH SIMULATION
C*** REPLICATION. FOR EACH PART TYPE, A RANDOM SAMPLE IS DRAWN
C*** FROM THE APPROPRIATE POISSON DISTRIBUTION TO DETERMINE THE
C*** NUMBER IN RESUPPLY AND THE NUMBER OF BACKORDERS IS COMPUTED
C*** USING THIS RESUPPLY NUMBER AND THE INITIAL STOCK
C*** LEVEL. THE MEAN OF THE POISSON FOR EACH PART IS THE
C*** PIPELINE FOR EACH TYPE. "NAC" INDICATES NUMBER OF ON-HAND
C*** AIRCRAFT AT THE START OF THE SIMULATION.
C***
C*** INPUTS --
C***   NAC        - CURRENT UE-STRENGTH.
C*** COMMON INPUTS --
C***   NPARTS     - NUMBER OF PART-TYPES BEING MODELED.
C***   RESUPP(K)  - (K=1,..,NPARTS) EXPECTED NUMBER OF TYPE-K PARTS
C***                IN RESUPPLY AT THE START OF THE SCENARIO. USED
C***                AS THE MEAN OF A POISSON DISTRIBUTION TO GENERATE
C***                A SAMPLE OF TYPE-K PARTS INITIALLY IN RESUPPLY.
C***   INITSJ(K)  - (K=1,....,NPARTS) INITIAL BASE STOCK LEVEL OF KTH
C***                PART TYPE.
C***   IQPA(K)    - (K=1,....,NPARTS) QPA OF KTH PART TYPE.
C***   BNRTS(K)   - (K=1,....,NPARTS) BASE-NOT-REPAIRED-THIS-STATION
C***                RATE. INDICATES PROPORTION OF TYPE-K FAILURES
C***                WHICH ARE REPAIRED AT THE BASE.
C*** COMMON OUTPUTS --
C***   NBACKO(K)  - (K=1,....,NPARTS) NUMBER OF BACKORDERS FOR KTH
C***                PART-TYPE. BACKORDERS ARE DEFINED AS
C***                (# IN RESUPPLY)-(INITIAL STOCK LEVEL)
C***   NBASE(K)   - (K=1,....,NPARTS) NUMBER OF TYPE-K PARTS IN BASE
C***                RESUPPLY.
C***   NDEPOT(K)  - (K=1,....,NPARTS) NUMBER OF TYPE-K PARTS IN DEPOT
C***                RESUPPLY.
C*************************************************************
C---
      PARAMETER MAXPRT=304
      COMMON/RSEED/ SEED
      COMMON /PARTS/ NPARTS,IQPA(MAXPRT),NBACKO(MAXPRT),
&        BRPRATE(MAXPRT),DRPRATE(MAXPRT),INITSJ(MAXPRT),RESUPP(MAXPRT),
&        BNRTS(MAXPRT),NBASE(MAXPRT),NDEPOT(MAXPRT)
C---
C---      *DO FOR(EACH PART TYPE)
           DO 200 K=1,NPARTS
C---
C---         *DRAW SAMPLE FROM POISSON DISTRIBUTION FOR NUMBER OF
C---                            PARTS IN RESUPPLY
              NRESUPP=IPOISSON(RESUPP(K),SEED)
C---
C---         *COMPUTE INITIAL BACKORDERS
              NBACKO(K)=NRESUPP - INITSJ(K)
```

```
C---
C---        *IF (IF BACKORDERS GREATER THAN PARTS ON-HAND)
             IF(NBACKO(K).LE.NAC*IQPA(K)) GOTO 100
C---
C---           *PRINT WARNING MESSAGE AND TRUNCATE NBACKO(K)
                WRITE(6,9001)
&               K,NBACKO(K),NAC,IQPA(K),NRESUPP,INITSJ(K),RESUPP(K)
                NBACKO(K)=NAC*IQPA(K)
                NRESUPP=NBACKO(K)+INITSJ(K)
C---
C---        *END IF(TRUNCATE BACKORDERS AT MAXIMUM AVAILABLE)
 100         CONTINUE
C---
C---        *ALLOCATE THESE PARTS BETWEEN BASE AND DEPOT RESUPPLY
             RBASE=0.0
             IF(BRPRATE(K).GT.0.0) RBASE=(1-BNRTS(K))/BRPRATE(K)
             RDEPOT=0.0
             IF(DRPRATE(K).GT.0.0) RDEPOT=BNRTS(K)/DRPRATE(K)
             NDEP=NBINUM(RDEPOT/(RBASE+RDEPOT),NRESUPP)
             NDEPOT(K)=NDEPOT(K)+NDEP
             NBASE(K)=NBASE(K)+NRESUPP-NDEP
C---
C---      *END DO (PARTS LOOP)
  200      CONTINUE
C---
   RETURN
 9001 FORMAT("0$$$$$$$$ INITBO ERROR - TOO MANY PARTS IN RESUPPLY",/,
& " $$$$$$$$     K=",I3," NBACKO(K)=",I5," NAC=",I3," IQPA(K)=",I3,
& /," $$$$$$$$     NRESUPP, INITSJ(K), RESUPP(K) = ",2I5,F10.3)
   END
```

```
C*****************************************************************
   SUBROUTINE INITPRT(IFILE,INFPART,SORTLGTH,PACBRK)
C*****************************************************************
C++ INITPRT    - LOAD AND INITIALIZE SPARE-PARTS DATA.
C***      THIS ROUTINE LOADS THE SPARE-PARTS INPUT DATA AND
C*** INITIALIZES THE STATISTICS AND TABLES NEEDED FOR
C*** SAMPLING TOTAL PART DEMANDS AND ALSO DETERMINING PART-TYPE
C*** FOR A GIVEN BROKEN PART.
C*****************************************************************
C---
      PARAMETER MAXAC=108,MAXBIT=36,MAXVEC=2+(MAXAC-1)/MAXBIT
      PARAMETER MAXPRT=304
      LOGICAL INFPART
      CHARACTER CNSN*18
      COMMON /PARTS/ NPARTS,IQPA(MAXPRT),NBACKO(MAXPRT),
&        BRPRATE(MAXPRT),DRPRATE(MAXPRT),INITSJ(MAXPRT),RESUPP(MAXPRT),
&        BNRTS(MAXPRT),NBASE(MAXPRT),NDEPOT(MAXPRT)
      COMMON /ALIASC/  FRACT(MAXPRT),IALIAS(MAXPRT),FPARTS
      COMMON /DEMAND/  ACMEAN, ACVAR, NPERAC
C---
C---  *IF(INFINITE PARTS ARE NOT ASSUMED, I.E. NORS AIRCRAFT ARE
C---        TO BE MODELED)THEN
        IF(INFPART)GO TO 900
C---
C---      *READ-IN PARTS DATA AND PERFORM ERROR CHECKS
          NPARTS=1
  100     CONTINUE
          READ(IFILE,END=200) CNSN,FRACT(NPARTS),IQPA(NPARTS),FAP,
&            INITSJ(NPARTS),RESUPP(NPARTS),BNRTS(NPARTS),BDAYS,DDAYS
             IF((FRACT(NPARTS).GT.0.0).AND.(FRACT(NPARTS).LE.1.0))
&                                                  GO TO 50
             IF(IQPA(NPARTS).GT.0)GO TO 50
             IF((FAP.GT.0.0).AND.(FAP.LE.1.0))GO TO 50
             IF(INITSJ(NPARTS).GE.0)GO TO 50
             IF(RESUPP(NPARTS).GE.0.0)GO TO 50
             IF((BNRTS(NPARTS).GE.0.0).AND.(BNRTS(NPARTS).LE.1.0))
&                                    GO TO 50
             IF(BDAYS.GE.0.0)GO TO 50
             IF(DDAYS.GE.0.0)GO TO 50
                WRITE(6,9003) CNSN,FRACT(NPARTS),IQPA(NPARTS),FAP,
&               INITSJ(NPARTS),RESUPP(NPARTS),BNRTS(NPARTS),BDAYS,DDAYS
                GO TO 100
  50         CONTINUE
             BRPRATE(NPARTS)=0.0
             IF(BDAYS.GT.0.0) BRPRATE(NPARTS)=1.0/(24.0*BDAYS)
             DRPRATE(NPARTS)=0.0
             IF(DDAYS.GT.0.0) DRPRATE(NPARTS)=1.0/(24.0*DDAYS)
             FRACT(NPARTS)=FRACT(NPARTS)*FAP*SORTLGTH
             IF(FRACT(NPARTS).LE.0.0)GO TO 100
             NPARTS=NPARTS+1
          IF(NPARTS.LE.MAXPRT)GO TO 100
             READ(IFILE,END=200)CNSN
             WRITE(6,9004)MAXPRT
```

```
 200    CONTINUE
        NPARTS=NPARTS-1
        WRITE(6,9005)NPARTS
C---
C---   *CLOSE-OUT SPARES INPUT FILE
        CALL FCLOSE(IFILE)
C---
C---   *COMPUTE MEAN AND VARIANCE OF RANDOM VARIABLE - TOTAL
C---        -PART-DEMANDS PER BROKEN AIRCRAFT
        CALL PSTAT(PACBRK,NPARTS,IQPA,FRACT,ACMEAN,ACVAR,NPERAC)
C---
C---   *CONVERT PART-DEMANDS-PER-FLYING-HOUR TO A PDF
        CALL MAKEPD(NPARTS,IQPA,FRACT)
C---
C---   *SET-UP TABLES NEEDED FOR ALIAS METHOD OF SAMPLING PART-DEMANDS
        CALL ALIAS(NPARTS,FRACT,IALIAS)
        FPARTS = FLOAT(NPARTS)
C---
C--- *ELSE (INFINITE PARTS ASSUMED)
     GO TO 950
 900 CONTINUE
C---
C---   *PRINT MESSAGE INDICATING INFINITE SPARE PARTS ASSUMPTION
        WRITE(6,9002)
C---
C--- *END IF (INFINITE SPARE-PARTS TEST)
 950 CONTINUE
C---
  RETURN
 9002 FORMAT(1H0,7X,"INFINITE SPARE PARTS ASSUMED FOR THIS SGM RUN, ",
&     /,"I.E., NO AIRCRAFT EVER WAITS FOR A SPARE PART.")
 9003 FORMAT("0$$$$$$$$ INITPRT ERROR - INVALID PART CHARACTERISTIC",
&  /,       " $$$$$$$$     NPARTS, CNSN = ",I3,1X,A18,
&  /,       " $$$$$$$$     FRACT, IQPA, FAP = ",F6.4,I4,F5.2,
&  /,       " $$$$$$$$     INITSJ, RESUPP, BNRTS = ",I4,2F8.3,
&  /,       " $$$$$$$$     BDAYS, UDAYS  = ",2F10.2)
 9004 FORMAT("0$$$$$$$$ INITPRT ERRUR - TOO MANY LRU TYPES",/,
&   " $$$$$$$$     MAXPRT = ",I5)
 9005 FORMAT(1H0,3X,"LRU TYPES - ",I4)
  END
```

```
C*****************************************************************
   SUBROUTINE INITREP
C*****************************************************************
C++ INITREP    - INITIALIZE VARIABLES FOR A SIMULATION REPLICATION.
C***      THIS ROUTINE PERFORMS THE LENGTHY INITIALIZATION NEEDED
C*** EACH SIMULATION REPLICATION OF THE SGM. THIS PROCESS
C*** IS ORGANIZED IN THE FOLLOWING MANNER. FIRST, MISCELLANEOUS
C*** OPERATIONS ARE PERFORMED, THEN SPARES INITIALIZATION, AND
C*** FINALLY, WORK CENTER INITIALIZATION. THE NUMEROUS COMMON OUTPUTS
C*** OF THIS ROUTINE ARE NOT DEFINED HERE, BUT THE
C*** OPERATIONS BEING PERFORMED SHOULD BE CLEAR FROM THE PROGRAM-
C*** DESIGN LANGUAGE (PDL) CORRESPONDING TO EACH OPERATION.
C*****************************************************************
C---
      PARAMETER  MAXAC=108,MAXWC=25,MAXBIT=36,MAXPRT=304,
&                 MAXVEC=2+(MAXAC-1)/MAXBIT
      PARAMETER LFLD=7,NPERWRD=MAXBIT/LFLD,MXINWC=1+(MAXAC-1)/NPERWRD
      PARAMETER MAXDAY=30,MAXCYC=10,MAXSTAT=5
      COMMON /ACSTATE/ LENGTH, NACVC(MAXVEC), IFLYVC(MAXVEC),
&                      MAINVC(MAXVEC), NORSVC(MAXVEC), LOSTVC(MAXVEC)
      COMMON /INPUT/   INITUE, NAC, PATTRIT, IRES, RNMCM, INFPART,
&                      MAXFLY(MAXCYC), INFMAN, ISCALE, IAUGMNT
      COMMON /PARTS/   NPARTS, IQPA(MAXPRT), NBACKO(MAXPRT),
&                      BRPRATE(MAXPRT), DRPRATE(MAXPRT), INITSJ(MAXPRT),
&                      RESUPP(MAXPRT), BNRTS(MAXPRT), NBASE(MAXPRT),
&                      NDEPOT(MAXPRT)
      COMMON /STATS/   EXPECT(MAXSTAT,MAXCYC,MAXDAY),
&                      NRESRV, IZDAY,ITOTRES(MAXDAY),LOSSTOT
      COMMON /WCBRK/   PACBRK, PACGABT, PBRKWC(MAXWC), PWCPROD,
&                      PBRKSEQ(2,MAXWC), INDXWC(MAXWC)
      COMMON /WCINPUT/ NWC, NCREWS(MAXWC), SRATE(MAXWC)
      COMMON /WCMAINT/ LISTRP(MXINWC,MAXWC), INREPR(MAXWC)
      LOGICAL INFPART
C---
C---  *INITIALIZE RESERVE AIRCRAFT COUNTS
       CALL ZERO(ITOTRES,MAXDAY)
       IZDAY=0
       NRESRV=0
C---
C---  *RESET INITIAL UE FOR THIS REPLICATION
       NAC = INITUE
       CALL UEUPDAT(NAC)
C---
C---  *INITIALIZE CUMULATIVE AIRCRAFT LOSSES TO NONE
       LOSSTOT = 0
C---
C---  *CLEAR AIRCRAFT-STATUS BIT-VECTORS
       CALL ZERO(LOSTVC,MAXVEC,
&                NORSVC,MAXVEC,
&                INREPR,MAXWC)
C---
C---  *SET FIRST NAC BITS OF THE FLYING BIT-VECTOR TO FLYABLE
       DO 100 I=1,LENGTH
```

```
 100     IFLYVC(I)=NACVC(I)
C---
C---  *IF(INFINITE PARTS NOT ASSUMED)THEN
       IF(INFPART)GO TO 200
C---
C---     *CLEAR BASE AND DEPOT RESUPPLY COUNTS
          CALL ZERO(NBASE,MAXPRT,NDEPOT,MAXPRT)
C---
C---     *CALCULATE INITIAL BACKORDERS FOR EACH PART-TYPE
          CALL INITBO(NAC)
C---
C---     *INITIALIZE NUMBER/DISTRIBUTION OF NORS AIRCRAFT
          NORS = NORSAC(NPARTS,IQPA,NBACKO)
          CALL TBITSL(NORS,IFLYVC,NORSVC)
C---
C---  *END IF (INFINITE SPARES TEST)
 200   CONTINUE
C---
C---  *INITIALIZE NUMBER/DISTRIBUTION OF AIRCRAFT IN MAINTENANCE
       NBRKAC = INT(RNMCM*FLOAT(NAC))
       CALL WCDIST(NBRKAC,PBRKSEQ,INDXWC,IFLYVC)
C---
      RETURN
      END
```

```
C*******************************************************************
      SUBROUTINE INITSCN(IFSCEN)
C*******************************************************************
C++ INITSCN    - READ AND INITIALIZES SCENARIO INPUTS.
C***      THIS ROUTINE LOADS THE SCENARIO PARAMETERS SPECIFIED
C*** BY THE USER. IT ALSO PREPARES THE SCRATCH FILE (FILE 03)
C*** WHICH IS USED TO WRITE A COPY OF THOSE SCENARIO
C*** PARAMETERS WHICH ARE ALLOWED TO VARY ON A DAILY BASIS
C*** THROUGHOUT THE SCENARIO. A LIST OF VALUES IS WRITTEN TO THIS
C*** SCRATCH FILE FOR EACH SIMULATION DAY. THEN, WHEN EACH
C*** FLYING DAY BEGINS (FOR EACH SIMULATION REPLICATION), THE
C*** PARAMETER VALUES FOR THAT DAY ARE LOADED.
C*******************************************************************
C---
      PARAMETER MAXWC=25, MAXVARY=5, MAXCYC=10
      CHARACTER*4 FTOTYM,LTOTYM
      CHARACTER*20 CHSEED
      CHARACTER*80 NEXTLINE
      LOGICAL VARY(MAXVARY),VARYSW,INFPART,INFMAN
      COMMON /RSEED/   SEED
      COMMON /TIME/ PREFLITE,SORTLGTH,WAITCYC,
&       TYMNITE,NSIM,ISIM,NUMDAY,IDAY,NCYCLES,ICYCLE
      COMMON /INPUT/  INITUE,NAC,PATTRIT,IRES,RNMCM,INFPART,
&         MAXFLY(MAXCYC),INFMAN,ISCALE,IAUGMNT
      COMMON /WCBRK/ PACBRK, PACGABT, PBRKWC(MAXWC), PWCPROD,
&                    PBRKSEQ(2,MAXWC), INDXWC(MAXWC)
C---
C--- *READ STORED INPUT
 5    FORMAT(V)
      READ(IFSCEN,5) (VARY(I),I=1,MAXVARY)
     READ(IFSCEN,5) CHSEED,FTOTYM,LTOTYM,INFMAN,INFPART,NSIM,NAC
&    ,PACBRK,RNMCM,NUMDAY,PREFLITE,SORTLGTH
     READ(IFSCEN,5) ISCALE
     NEXTLINE=' '
     INITUE=NAC
     WRITE(06,9000) NSIM,CHSEED,NAC
     VARYSW=.F.
     READ(IFSCEN,5) PATTRIT,PACGABT,NCYCLES,IRES,WAITCYC,TYMNITE
     READ(IFSCEN,5) (MAXFLY(I),I=1,NCYCLES)
     IF (VARY(4)) GOTO 100
        WRITE(06,9008) IRES
        GOTO 200
 100  CONTINUE
        WRITE(06,9007)
        CALL CONCAT(NEXTLINE,28,'VARY BY DAY',1,11)
        VARYSW=.T.
 200  CONTINUE
     IF (VARY(5)) GOTO 300
        WRITE(06,9010) MAXFLY(1)
        GOTO 400
 300  CONTINUE
        WRITE(06,9009)
        VARYSW=.T.
```

```
          CALL CONCAT(NEXTLINE,45,'VARIES BY CYCLE/DAY',1,19)
  400  CONTINUE
        WRITE(06,9011) NEXTLINE
        IF (VARY(3)) GOTO 500
           WRITE(06,9012) NUMDAY,NCYCLES,FTOTYM,LTOTYM,PREFLITE,SORTLGTH,
&          WAITCYC,TYMNITE
           GOTO 600
  500  CONTINUE
          WRITE(06,9013) NUMDAY,FTOTYM,LTOTYM,PREFLITE,SORTLGTH
          VARYSW=.T.
  600  CONTINUE
      WRITE(06,9014) RNMCM,PACBRK
      NEXTLINE=' '
      IF (VARY(1)) GOTO 700
          WRITE(06,9015) PATTRIT
          GOTO 800
  700  CONTINUE
          WRITE(06,9016)
          VARYSW=.T.
          CALL CONCAT(NEXTLINE,19,'BY DAY',1,6)
  800  CONTINUE
      IF (VARY(2)) GOTO 900
          WRITE(06,9017) PACGABT,NEXTLINE
          GOTO 1000
  900  CONTINUE
          CALL CONCAT(NEXTLINE,46,'BY DAY',1,6)
          WRITE(06,9018) NEXTLINE
          VARYSW=.T.
  1000  CONTINUE
       DO 1200 IDAY=1,NUMDAY
          WRITE(03) PATTRIT,PACGABT,NCYCLES,IRES,WAITCYC,TYMNITE
          WRITE(03) (MAXFLY(J),J=1,NCYCLES)
              IF (.NOT.VARYSW) GOTO 1100
                  WRITE(06,9001) 'DAY =',IDAY
                  IF (VARY(1)) WRITE(06,9002) PATTRIT
                  IF (VARY(2)) WRITE(06,9003) PACGABT
                  IF (VARY(3)) WRITE(06,9004) NCYCLES,WAITCYC,TYMNITE
                  IF (VARY(4)) WRITE(06,9005) IRES
                  IF (VARY(5)) WRITE(06,9006) 'CYCLE',(J,J=1,NCYCLES)
                  IF (VARY(5)) WRITE(06,9006) 'MAX-FLY',
&                    (MAXFLY(J),J=1,NCYCLES)
  1100    CONTINUE
        IF (IDAY.EQ.NUMDAY) GOTO 1200
            READ(IFSCEN,5) PATTRIT,PACGABT,NCYCLES,IRES,WAITCYC,TYMNITE
            READ(IFSCEN,5) (MAXFLY(J),J=1,NCYCLES)
  1200  CONTINUE
C--
C--   *CONVERT USER SEED TO A REAL NUMBER
        DECODE(CHSEED,5)SEED
C--
C--   *CLOSE SCENARIO INPUT FILE
C--    CALL FCLOSE(IFSCEN)
C--
```

```
  RETURN
 9001   FORMAT('0',A5,I3)
 9002   FORMAT(' ATTRITION RATE =',F6.4)
 9003   FORMAT(' GROUND-ABORT RATE =',F6.4)
 9004   FORMAT(' WAVES PER DAY =',I3/' WAIT TIME =',F4.2,
&          /' OVERNITE RECOVERY =',F5.2)
 9005   FORMAT(' RESERVES =',I3)
 9006   FORMAT(' ',A7,10(2X,I3))
C
 9000  FORMAT('1'///4X'******************************************',
&       '****************************',/,4X'*************************'
&       ,'*****  SGM RUN  *****************************',/,4X
&       '***************************************************************',
&       '*******',///,4X
&       'SIMULATION -   REPLICATIONS =',I4,5X,
&       'RANDOM NUMBER SEED = ',A8/'0',3X,'AIRCRAFT -',3X,
&       'UE =',I3)
 9007 FORMAT('+',28X,'RESERVES')
 9008 FORMAT('+',27X,'RESERVES =',I3)
 9009 FORMAT('+',44X,'MAXIMUM LAUNCH-SIZE')
 9010 FORMAT('+',44X,'MAXIMUM LAUNCH-SIZE =',I3)
 9011 FORMAT(' ',A80/'0',3X,'FLYING SCHEDULE -'/'0',10X,'WAVES',3X,
&       'TAKEOFF',
&       ' TIMES',6X,'MINIMAL',3X,'SORTIE',2X,'WAIT',2X,'OVERNIGHT'/
&       4X,'DAYS',2X,'PER DAY',3X,'FIRST',3X,'LAST',4X,
&       'TURNAROUND',2X,'LENGTH',2X,'TIME',2X,'RECOVERY')
 9012 FORMAT('0',3X,I3,5X,I2,6X,A4,4X,A4,6X,F5.2,5X,F5.2,3X,F4.2,
&       3X,F5.2)
 9013 FORMAT('0',3X,I3,4X,'VARY',5X,A4,4X,A4,6X,F5.2,5X,F5.2,2X,
&       'VARIES',2X,'VARIES'/' ',9X,'BY DAY',39X,'BY DAY',2X,'BY DAY')
 9014  FORMAT('0',3X,'RATES -'/'0',6X,'INITIAL',17X,'AIRCRAFT'/
&       6X,'NMCM RATE',3X,'ATTRITION',3X,'BREAK RATE',3X,
&       'GROUND-ABORT'/'0',7X,F5.3,7X,13X,F6.4,)
 9015 FORMAT('+',17X,F6.2)
 9016 FORMAT('+',18X,'VARIES')
 9017 FORMAT('+',43X,F6.4/' ',A80)
 9018 FORMAT('+',45X,'VARIES'/' ',A80)
  END
```

```
*************************************************************
   SUBROUTINE INITWC(IFILE,INFMAN,NAC)
C***********************************************************************
C** INITWC     - LOAD AND INITIALIZE MAINTENANCE WORK CENTER DATA.
C***     THIS ROUTINE INITIALIZES THE INFORMATION NEEDED
C*** FOR MODELING THE AIRCRAFT MAINTENANCE WORK-CENTERS.
C*** IT READS THE MAINTENANCE MANPOWER INPUT FILE, PRINTS
C*** A LISTING OF THESE INPUTS, AND COMPUTES ADJUSTED BREAK-
C*** RATE ARRAYS FOR WORK-CENTER BREAKS
C***
C*** INPUTS --
C***      IFILE    - UNIT NUMBER OF THE INPUT FILE FROM
C***                 WHICH THE WORK-CENTER INPUTS ARE READ.
C***                 THIS FILE IS CLOSED-OUT AFTER THE INPUTS
C***                 ARE READ
C***      INFMAN   - LOGICAL VARIABLE INDICATING WHETHER INFINITE
C***                 MANPOWER IS ASSUMED FOR ALL WORK-CENTERS. IF
C***                 INFMAN=TRUE THEN NUMBER OF SERVERS FOR EACH WC
C***                 IS SET EQUAL TO THE MAXIMUM ALLOWABLE NUMBER
C***                 OF AIRCRAFT.
C***      NAC      - UE (UNIT EQUIPMENT); NUMBER OF AIRCRAFT
C***                 POSSESSED BY THE BASE OF INTEREST.
C*** COMMON INPUTS --
C***      PACBRK   - USER-INPUT AIRCRAFT BREAK-RATE.  PROBABILITY
C***                 THAT AN AIRCRAFT RETURNING FROM A SORTIE REQUIRES
C***                 UNSCHEDULED MAINTENANCE IN AT LEAST 1 WORK-CENTER.
C***      PACGABT  - PROBABILITY THAT AN AIRCRAFT GROUND-ABORTS DURING
C***                 THE PRE-TAKEOFF PERIOD.
C*** COMMON OUTPUTS --
C***      NWC      - NUMBER OF WORK-CENTERS TO BE MODELED
C***      NCREWS   - NUMBER-OF-SERVERS ARRAY. NCREWS(I) IS THE
C***                 NUMBER OF SERVERS IN THE ITH WORK-CENTER.
C***      SRATE    - SERVICE-RATES ARRAY. SRATE(I) IS THE SERVICE-
C***                 RATE (IN AIRCRAFT PER HOUR) OF THE SERVERS
C***                 FOR THE ITH WORK-CENTER.
C***      PBRKSEQ  - WORK-CENTER BREAK ARRAYS FOR THE SEQUENTIAL
C***                 SAMPLING PROCESS OF DETERMINING WHICH WORKCENTERS
C***                 AIRCRAFT BREAK INTO.
C***      INDXWC   - SORTED ARRAY OF WORK-CENTER INDICES USED WITH
C***                 SEQUENTIAL-SAMPLING PROCESS.
C***********************************************************************
C--
      PARAMETER MAXWC=25, MAXAC=108, LFLD=7
      COMMON /WCINPUT/ NWC, NCREWS(MAXWC), SRATE(MAXWC)
      COMMON /WCBRK/ PACBRK, PACGABT, PBRKWC(MAXWC), PWCPROD,
     &               PBRKSEQ(2,MAXWC), INDXWC(MAXWC)
      LOGICAL INFMAN
C--
C--  *READ AND ECHO-PRINT MAINTENANCE MANPOWER INPUT FILE
      CALL WCREAD(IFILE,MAXWC,NWC,PBRKWC,NCREWS,SRATE)
C--
C--  *IF INFINITE MANPOWER ASSUMED -- RESET NUMBER OF SERVERS
C--   PER SHIFT TO MAX NUMBER OF AIRCRAFT; THUS NO AIRCRAFT
```

```
C---   WILL EVER WAIT FOR A SERVER
       IF(INFMAN) CALL SPRAY(MAXAC,NCREWS,NWC)
       IF(INFMAN)WRITE(6,9001)
C---
C---  *INITIALIZE WORK-CENTER BREAK ARRAYS FOR GROUND-ABORTS AND
C---   BREAKS; NOTE THAT THE SAME ARRAYS ARE CURRENTLY USED FOR BOTH
       CALL WCPROB(NWC,PBRKWC,PBRKSEQ,INDXWC,PWCPROD)
C---
C---  *PERFORM ERROR CHECK TO ENSURE LFLD PARAMETER LARGE ENOUGH
C---         SO THAT THE BIT-FIELD CAN STORE THE MAX AC #
       IF(MAXAC.GT.2**LFLD)WRITE(6,9002)LFLD,MAXAC
C---
   RETURN
 9001 FORMAT(1H0,7X,"INFINITE MANPOWER ASSUMED FOR THIS SGM RUN -",/,
&   7X,"I.E., THERE ARE NEVER ANY AIRCRAFT QUEUES IN MAINTENANCE.")
 9002 FORMAT("0$$$$$$$$ INITWC ERROR - LFLD PARAMETER TOO SMALL",/,
&           " $$$$$$$$     LFLD, MAXAC = ",2I5)
   END
```

```
C*****************************************************************
   INTEGER FUNCTION IPOISSON(RMEAN,SEED)
C*****************************************************************
C** IPOISSON   - GENERATE RANDOM SAMPLE FROM A POISSON DISTRIBUTION.
C***      THIS ROUTINE GENERATES A RANDOM SAMPLE FROM A
C*** POISSON DISTRIBUTION WITH A GIVEN MEAN. THE EXPONENTIAL-DRAW
C*** METHOD IS USED FOR DISTRIBUTIONS WITH SMALL MEANS, AND
C*** A NORMAL APPROXIMATION IS USED FOR LARGER MEANS ( >20).
C***
C*** INPUT --
C***   RMEAN      - MEAN OF POISSON DISTRIBUTION FROM WHICH SAMPLE
C***                IS TO BE GENERATED.
C*** INPUT/OUTPUT --
C***   SEED       - SEED OF RANDOM NUMBER GENERATOR.
C*****************************************************************
C--
C--      *IF(INPUT PARAMETER IS A LEGITIMATE MEAN FOR A POISSON)
          IF(RMEAN.LT.0.0)GO TO 400
C--
C--        *IF(MEAN IS NOT TOO LARGE)
             IF(RMEAN .GT. 20.0) GO TO 200
C--
C--          *USE EXPONENTIAL DRAW METHOD FOR POISSON SAMPLE
               IPOISSON=-1
               PROD=1.0
               TEST=EXP(-RMEAN)
  100          CONTINUE
                  IPOISSON=IPOISSON+1
                  PROD=PROD*UNIFM1(SEED)
               IF(PROD.GE.TEST)GO TO 100
C--
C--        *ELSE (LARGE MEAN)
             GO TO 300
  200        CONTINUE
C--
C--          *USE NORMAL APPROXIMATION TO POISSON
               IPOISSON=MAX0(0, INT(XNORM(RMEAN,SQRT(RMEAN),SEED)+.5))
C--
C--        *END IF (SIZE OF MEAN TEST)
  300        CONTINUE
C--
C--      *ELSE (MEAN IS LESS THAN ZERO)
          GO TO 500
  400     CONTINUE
C--
C--        *SET RETURN VALUE TO ZERO AND PRINT ERROR MESSAGE
             IPOISSON = 0
             WRITE(6,9001)RMEAN
C--
C--      *END IF (LEGITIMATE MEAN TEST)
  500     CONTINUE
C--
   RETURN
```

```
 9001 FORMAT("0$$$$$$$$$ IPOISSON ERROR - NEGATIVE MEAN ",/,
&            " $$$$$$$$$     RMEAN = ",F10.5)
      END
```

```
C*************************************************************
   INTEGER FUNCTION LBITS(IWORD,NBITS)
C*************************************************************
C** LBITS     - MASK-OFF LEFTMOST 1-BITS IN A COMPUTER WORD.
C***      LBITS IS A FORTRAN FUNCTION WHICH WILL SELECT A GIVEN
C*** NUMBER OF 1-BITS FROM THE LEFTMOST PORTION OF A GIVEN INPUT
C*** WORD. LBITS RETURNS A WORD CONSISTING OF THESE SELECTED
C*** 1-BITS WITH 0'S EVERYWHERE ELSE. NOTE THAT THE INPUT
C*** WORD SHOULD CONTAIN AT LEAST AS MANY 1-BITS AS THE NUMBER TO BE
C*** SELECTED, 'NBITS'.
C***      THIS ROUTINE IS SPECIFIC TO A COMPUTER WITH 36-BIT WORDS
C*** SINCE IT WORKS BY EXTRACTING 6-BIT FIELDS.
C***
C*** INPUTS --
C***   IWORD      - WORD FROM WHICH THE 1-BITS ARE TO BE SELECTED.
C***                THIS WORD SHOULD CONTAIN AT LEAST AS MANY 1-BITS
C***                AS REQUESTED. IF MORE THAN THAT ARE REQUESTED,
C***                THIS ROUTINE WILL RETURN AN EXACT COPY OF THE INPUT.
C***   NBITS      - NUMBER OF 1-BITS TO BE SELECTED FROM 'IWORD'.
C*** OUTPUT --
C***   LBITS      - A COPY OF THE PORTION OF THE INPUT WORD CONTAINING
C***                THE SPECIFIED NUMBER OF LEFTMOST 1-BITS.
C*** COMMON TABLES USED --
C***   ICOUNT(I)  - NUMBER OF 1-BITS IN THE BINARY REPRESENTATION OF THE
C***                INDEX I.   I=0,1,2,...,63
C***   MSKLFT(I)  - MASK FOR WHICH THE LEFTMOST I-BITS ARE 1-BITS,
C***                AND THE REMAINDER OF THE WORD IS ZERO.  I=1,2,3,...,36
C***   MASK(I)    - CONTAINS A 1 IN THE ITH BIT (COUNTING FROM THE LEFT)
C***                AND 0'S EVERYWHERE ELSE.  I=0,1,2,...,35
C*************************************************************
C---
      PARAMETER  MAXBIT=36,LFIELD=6
      COMMON /BITS/ MASK0,MASK(35), MLEFT0,MSKLFT(36),
&                              IZCOUT,ICOUNT(63)
C---
C---       *IF(NO BITS ARE REQUESTED)THEN
            IF(NBITS.GT.0) GO TO 1000
C---
C---         *RETURN A VECTOR OF ALL ZEROES
              LBITS = 0
C---
C---       *ELSE (SELECT LEFTMOST BITS)
            GO TO 5000
 1000       CONTINUE
C---
C---         **SEARCHING FROM LEFT TO RIGHT, FIND THE 6-BIT FIELD
               CONTAINING THE LAST BIT TO BE SELECTED
C---
C---           *INITIALIZE DO
                IBIT   = 0
                IFOUND = 0
C---
C---           *DO UNTIL(APPROPRIATE 6-BIT FIELD IS FOUND)
```

```
 2000          CONTINUE
C--
C--              *UPDATE NUMBER OF 1-BITS FOUND SO FAR
                  IFOUND = IFOUND + ICOUNT(FLD(IBIT,LFIELD,IWORD))
C--
C--              *UPDATE FIELD COUNTER
                  IBIT = IBIT + LFIELD
C--
C--            *END DO (FIELD LOOP)
                IF((IFOUND.LT.NBITS) .AND. (IBIT.LT.MAXBIT)) GO TO 2000
C--
C--          *COMPUTE NUMBER OF EXTRA 1-BITS INCLUDED
              NEXTRA = IFOUND - NBITS
C--
C--          *PERFORM ERROR CHECK TO ENSURE PROPER NUMBER OF 1S FOUND
              IF(NEXTRA.LT.0)WRITE(6,9001)NBITS,IFOUND,NEXTRA
C--
C--          *DO WHILE(THERE ARE EXTRA 1'S TO ELIMINATE)
 3000         CONTINUE
              IF(NEXTRA.LE.0) GO TO 4000
C--
C--            *DECREMENT BIT COUNTER
                IBIT = IBIT - 1
C--
C--            *DECREMENT EXTRA-BIT COUNTER IF THIS BIT IS A 1
                IF(AND(IWORD,MASK(IBIT)).NE.0) NEXTRA=NEXTRA-1
C--
C--          *END DO (EXTRA 1'S LOOP)
              GO TO 3000
 4000         CONTINUE
C--
C--          *RETURN PORTION OF INPUT UP TO, BUT NOT INCLUDING
C--                                                 THIS LAST BIT
              LBITS = AND(IWORD,MSKLFT(IBIT))
C--
C--        *END IF (ZERO BITS REQUESTED TEST)
 5000       CONTINUE
C--
   RETURN
 9001 FORMAT("0$$$$$$$$ LBITS ERROR - TOO FEW 1-BITS TO MASK",/,
&           " $$$$$$$$     NBITS, IFOUND, NEXTRA = ",3I5)
   END
```

```
C******************************************************************
   SUBROUTINE MAKEPD (N,IQPA,DEMAND)
C******************************************************************
C** MAKEPD     - CONVERTS PARTS DEMAND ARRAY INTO A PDF.
C***     THE ALIAS METHOD REQUIRES A LEGITIMATE PROBABILITY
C***     DISTRIBUTION AS AN INPUT. THIS ROUTINE TAKES THE QPA
C***     AND PROB[DEMAND] FIGURES FOR EACH PART, AND FORMS A PDF
C***     FROM THEIR PRODUCTS. THERE ARE N PART TYPES.
C******************************************************************
C---
      DIMENSION IQPA(N),DEMAND(N)
C---
      SUM = 0.0
      DO 30 K=1,N
         DEMAND(K) = DEMAND(K) * FLOAT(IQPA(K))
         SUM = SUM + DEMAND(K)
  30  CONTINUE
      RECIP = 1.0 / SUM
      SUM = 0.0
      DO 60 K=1,N-1
         DEMAND(K) = DEMAND(K) * RECIP
         SUM = SUM + DEMAND(K)
  60  CONTINUE
      DEMAND(N) = 1.0 - SUM
C---
   RETURN
   END
```

```
C*****************************************************************
   INTEGER FUNCTION MNOM (DUMMY)
C*****************************************************************
C++ MNOM       - GENERATE MULTINOMIAL SAMPLE FOR PART DEMAND TYPE.
C***    THIS FUNCTION GENERATES A MULTINOMIAL SAMPLE INDICATING
C***    WHICH PART TYPE HAS BROKEN.  IT USES TWO TABLES CREATED
C***    PREVIOUSLY BY SUBROUTINE 'ALIAS'.
C***
C*** COMMON INPUTS -
C***        FPARTS    - FLOATING-POINT VALUE OF NUMBER OF PART TYPES, N.
C***        FRACT(I)  - TABLE OF FRACTIONAL CUTOFF VALUES USED BY THE
C***                    ALIAS METHOD.    I=1,2,...,N
C***        IALIAS(I) - TABLE OF ALIASES USED BY ALIAS METHOD. I=1,...,N
C*** OUTPUT -
C***        MNOM      - INDEX INDICATING TYPE OF PART WHICH HAS BROKEN.
C***                    MNOM=1,2,...,N  (NUMBER OF PART TYPES)
*************************************************************
C---
      PARAMETER MAXPRT=304
      COMMON /RSEED/ SEED
      COMMON /ALIASC/ FRACT(MAXPRT),IALIAS(MAXPRT),FPARTS
C---
C---  *MAKE 'U' A UNIFORM (0,N) RANDOM REAL NUMBER
       U = FPARTS * UNIFM1(SEED)
C---  *
C---  *MAKE 'IU' A UNIFORM RANDOM INTEGER (1,N)
       IU = IFIX(U) + 1
C---
C---  *IF NECESSARY, REPLACE 'IU' BY ITS ALIAS
       IF (U .GT. FRACT(IU))  IU = IALIAS(IU)
C---
C---  RESULT IS RETURNED AS 'MNOM'
       MNOM = IU
C---
   RETURN
   END
```

```
C******************************************************************
   SUBROUTINE MUPDATE(NWC,MAINVC)
C******************************************************************
C** MUPDATE    - UPDATE MAINTENANCE AIRCRAFT-STATE BIT-VECTOR.
C***   MUPDATE UPDATES THE OVERALL MAINTENANCE BIT-VECTOR. THIS
C*** UPDATING PROCESS CONSISTS OF GOING THROUGH EACH WORK-CENTER
C*** LIST OF AIRCRAFT AND MARKING THE CORRESPONDING BIT-POSITION IN
C*** THE MAINTENANCE BIT-VECTOR FOR ANY AIRCRAFT IN SUCH A LIST; THUS,
C*** ANY AIRCRAFT IN AT LEAST ONE WORK-CENTER LIST WILL BE MARKED AS
C*** BEING IN MAINTENANCE STATUS.
C***    THE MAINTENANCE BIT-VECTOR IS UPDATED AT THE BEGINNING OF
C*** EACH FLYING CYCLE.  IT IS NOT MAINTAINED DURING THE FLYING CYCLE,
C*** SINCE IT IS ONLY NEEDED AT THE START OF PREFLIGHT TO DETERMINE
C*** THOSE AIRCRAFT WHICH ARE NOT MISSION-CAPABLE BECAUSE THEY ARE
C*** IN AT LEAST ONE WORK-CENTER. IT IS MUCH FASTER TO UPDATE
C*** ONCE EACH FLYING CYCLE RATHER THAN UPDATING IT EACH TIME A
C*** WORK-CENTER BREAK OR REPAIR OCCURS.
C***
C*** INPUTS -
C***    NWC         - NUMBER OF WORK-CENTERS BEING SIMULATED
C*** COMMON INPUTS -
C***    MASK(I)     - CONTAINS A 1 IN THE ITH BIT (COUNTING FROM LEFT)
C***                  AND ZEROES EVERYWHERE ELSE. I=0,...,35
C***    LENGTH      - LENGTH (IN WORDS) OF AIRCRAFT BIT-VECTORS
C***    INREPR(J)   - NUMBER OF AIRCRAFT IN WORKCENTER-J.
C***    LISTRP(I,J) - LISTRP( , ,J) IS A LIST OF AIRCRAFT NUMBERS
C***                  INDICATING THOSE AIRCRAFT REQUIRING MAINTENANCE IN
C***                  THE JTH WORK-CENTER (J=1,2,...,NWC). THIS LIST
C***                  CONTAINS EXACTLY INREPR(J) AIRCRAFT NUMBERS. TO SAVE
C***                  SPACE, THESE LISTS HAVE BEEN PACKED INTO BIT-FIELDS
C***                  INSTEAD OF WORDS. EACH NUMBER IS STORED IN A BIT-FIELD
C***                  "LFLD" BITS WIDE; HENCE, IF "MAXBIT" IS THE LENGTH
C***                  OF A COMPUTER WORD ON THIS SYSTEM, THEN THERE ARE
C***                  (MAXBIT/LFLD) BIT-FIELDS STORED PER WORD. THE AIRCRAFT
C***                  NUMBERS STORED IN THESE BIT-FIELDS INDICATE A UNIQUE
C***                  BIT-POSITION IN THE VARIOUS AIRCRAFT-STATUS BIT-
C***                  VECTORS. THE AIRCRAFT ARE NUMBERED,LEFT-TO-RIGHT,
C***                  0,1,2,...,(MAXAC-1) . TO GET THE ITH AIRCRAFT NUMBER
C***                  IN A WORK-CENTER LIST, THE CORRESPONDING
C***                  BIT-POSITION AND WORD-INDEX MUST BE COMPUTED.
C*** OUTPUTS -
C***    MAINVC      - MAINTENANCE AIRCRAFT-STATUS BIT-VECTOR. EACH BIT
C***                  REPRESENTS AN AIRCRAFT. A 1 INDICATES THE
C***                  CORRESPONDING AIRCRAFT IS BEING REPAIRED IN AT
C***                  LEAST ONE WORK-CENTER, AND 0 INDICATES THE
C***                  AIRCRAFT IS NOT CURRENTLY IN MAINTENANCE.
C******************************************************************
C---
      PARAMETER MAXWC=25
      PARAMETER MAXAC=108,MAXBIT=36,MAXVEC=2+(MAXAC-1)/MAXBIT
      PARAMETER LFLD=7,NPERWRD=MAXBIT/LFLD,MXINWC=1+(MAXAC-1)/NPERWRD
      COMMON /WCMAINT/ LISTRP(MXINWC,MAXWC), INREPR(MAXWC)
      COMMON /BITS/ MASK0,MASK(35),MLEFT0,MSKLFT(36),
&                    IZCOUT, ICOUNT(63)
      DIMENSION MAINVC(MAXVEC)
C---
C---  *INITIALIZE MAINTENANCE BIT-VECTOR TO NO AIRCRAFT
       CALL ZERO(MAINVC,MAXVEC)
C---
C---  *DO FOR(EACH WORK-CENTER)
       IF(NWC.EQ.0) GO TO 400
       DO 300 J=1,NWC
C---
C---     *DO WHILE(STILL AIRCRAFT IN THIS WORK-CENTER MAINTENANCE LIST)
          NUM = INREPR(J)
          IF(NUM.EQ.0) GO TO 200
          DO 100 I=1,NUM
C---
C---        *GET NEXT AIRCRAFT NUMBER ON LIST FROM APPROPIATE BIT-FIELD
             IAC = FLD(MOD(I-1,NPERWRD)*LFLD,LFLD,
&                          LISTRP(1+(I-1)/NPERWRD,J))
C---
C---        *COMPUTE WORD AND BIT POSITIONS INDICATED BY THIS AC #
             IWORD = 2 + IAC/MAXBIT
             IBIT  = MOD(IAC,MAXBIT)
C---
C---        *MARK CORRESPONDING POSITION IN AIRCRAFT MAINT BIT-VECTOR
             MAINVC(IWORD) = OR(MAINVC(IWORD), MASK(IBIT))
C---
C---     *END DO (WORK-CENTER LIST LOOP)
  100     CONTINUE
  200     CONTINUE
C---
C---  *END DO (WORK-CENTER LOOP)
  300  CONTINUE
  400  CONTINUE
C---
C---  *COMPUTE TOTAL AIRCRAFT IN MAINTENANCE BY COUNTING 1-BITS IN
C---                    AIRCRAFT MAINTENANCE VECTOR
       MAINVC(1) = N1VECT(MAINVC)
C---
     RETURN
     END
```

```
C*********************************************************************
   INTEGER FUNCTION N1BITS(IWORD)
C*********************************************************************
C** N1BITS     - COUNT NUMBER OF 1-BITS IN A COMPUTER WORD.
C***     N1BITS IS A FORTRAN SUBROUTINE WHICH WILL RETURN THE
C*** NUMBER OF 1-BITS IN A GIVEN WORD.  THIS ROUTINE IS SPECIFIC
C*** TO A COMPUTER WITH 36-BIT WORDS, SINCE IT WORKS BY EXTRACTING
C*** 6-BIT FIELDS FROM THE WORD. IT USES EACH 6-BIT FIELD EXTRACTED
C*** AS AN INDEX INTO A TABLE, AND THE ENTRIES IN THE TABLE CONTAIN THE
C*** CORRESPONDING NUMBER OF 1-BITS FOR THAT INDEX.
C***
C*** INPUT --
C***   IWORD      - WORD FOR WHICH THE 1-BITS ARE TO BE COUNTED
C*** OUTPUT --
C***   N1BITS     - NUMBER OF 1-BITS IN THE GIVEN INPUT WORD. HENCE,
C***                N1BITS RETURNS AN INTEGER BETWEEN 0 AND 36.
C*** TABLE USED -
C***   ICOUNT(I) - NUMBER OF 1-BITS IN THE BINARY REPRESENTATION OF THE
C***                INDEX I.  I=1,...,63
C*********************************************************************
C---
     COMMON /BITS/    MASK0,MASK(35), MLEFT0,  MSKLFT(36),
&                                IZCOUT,ICOUNT(63)
C---
           N1BITS = ICOUNT( FLD( 0,6,IWORD) )
&                 + ICOUNT( FLD( 6,6,IWORD) )
&                 + ICOUNT( FLD(12,6,IWORD) )
&                 + ICOUNT( FLD(18,6,IWORD) )
&                 + ICOUNT( FLD(24,6,IWORD) )
&                 + ICOUNT( FLD(30,6,IWORD) )
C---
   RETURN
   END
```

```
C*************************************************************
      INTEGER FUNCTION N1VECT(IARRAY)
C*************************************************************
C** N1VECT     - COUNT NUMBER OF 1-BITS IN A BIT-VECTOR.
C***      N1VECT IS A FORTRAN SUBROUTINE WHICH WILL RETURN THE
C*** NUMBER OF 1-BITS IN THE WORDS COMPRISING A GIVEN INPUT ARRAY,
C*** NOT INCLUDING THE FIRST WORD. THIS ROUTINE IS USED TO COUNT
C*** THE NUMBER OF 1-BITS IN THE VARIOUS AIRCRAFT-STATUS
C*** BIT-VECTORS.
C***      THIS ROUTINE IS SPECIFIC TO A 36-BIT-WORD COMPUTER, SINCE IT
C*** WORKS BY EXTRACTING 6-BIT FIELDS FROM THE ARRAY WORDS. IT USES
C*** EACH 6-BIT FIELD EXTRACTED AS AN INDEX INTO A TABLE, AND THE
C*** ENTRIES IN THE TABLE CONTAIN THE CORRESPONDING NUMBER OF 1-BITS
C*** FOR THAT INDEX.
C***
C*** INPUT -
C***   IARRAY    - ARRAY FOR WHICH THE 1-BITS ARE TO BE COUNTED.
C*** COMMON TABLE USED -
C***   ICOUNT(I) - NUMBER OF 1-BITS IN THE BINARY REPRESENTATION OF THE
C***               INDEX I.  I=1,...,63
C*** OUTPUT -
C***   N1VECT    - NUMBER OF 1-BITS IN THE GIVEN INPUT ARRAY,
C***               EXCLUDING THE FIRST WORD.
C*************************************************************
C---
      PARAMETER MAXAC=108,MAXBIT=36,MAXVEC=2+(MAXAC-1)/MAXBIT
      COMMON /BITS/ MASK0,MASK(35), MLEFT0,MSKLFT(36),
&                                 IZCOUT,ICOUNT(63)
      COMMON /ACSTATE/ LENGTH,NACVC(MAXVEC), IFLYVC(MAXVEC),
&                MAINVC(MAXVEC),NORSVC(MAXVEC), LOSTVC(MAXVEC)
      DIMENSION IARRAY(1)
C---
         N1VECT = 0
         DO 1000  I=2,LENGTH
            IWORD  = IARRAY(I)
            N1VECT = N1VECT + ICOUNT( FLD( 0,6,IWORD) )
&                           + ICOUNT( FLD( 6,6,IWORD) )
&                           + ICOUNT( FLD(12,6,IWORD) )
&                           + ICOUNT( FLD(18,6,IWORD) )
&                           + ICOUNT( FLD(24,6,IWORD) )
&                           + ICOUNT( FLD(30,6,IWORD) )
 1000    CONTINUE
C---
      RETURN
      END
```

```
C*****************************************************************
   INTEGER FUNCTION NBINOM(PBINOM,NTRYS)
C*****************************************************************
C** NBINOM     - GENERATE RANDOM SAMPLE FROM BINOMIAL DISTRIBUTION.
C***      NBINOM GENERATES A RANDOM SAMPLE FROM A BINOMIAL DISTRIBUTION
C*** WITH THE GIVEN INPUT CHARACTERISTICS. THIS ROUTINE USES A
C*** COMBINATION OF TWO METHODS TO GENERATE THIS SAMPLE. FOR
C*** BINOMIALS WITH RELATIVELY SMALL NUMBERS OF TRIALS, THE
C*** STRAIGHTFORWARD BERNOULLI TRIALS METHOD IS USED. FOR LARGER
C*** VALUES, THE INVERSE TRANSFORM METHOD IS USED.
C***      NOTE THAT THE NUMBER OF FAILURES IN A BINOMIAL SAMPLE IS
C*** THE COMPLEMENT OF THE NUMBER OF SUCCESSES IN THAT DRAW. HENCE
C*** THIS ROUTINE WILL SAMPLE FROM THE COMPLEMENTARY BINOMIAL
C*** DISTRIBUTION OF FAILURES WHEN THE PROBABILITY OF SUCCESS IS
C*** GREATER THAN .5 .
C***
C*** INPUTS --
C***   PBINOM    - PROBABILITY CHARACTERISTIC OF THE BINOMIAL.
C***               PBINOM ALSO EQUALS THE PROBABILITY THAT THE
C***               BERNOULLI VARIABLE UNDERLYING THIS BINOMIAL EQUALS 1.
C***   NTRYS     - NUMBER OF BERNOULLI TRIALS CHARACTERIZING THIS
C***               BINOMIAL.
C*****************************************************************
C---
      COMMON /RSEED/ SEED
C---
C---  *INITIALIZE SAMPLE TO NO SUCCESSES
       NBINOM = 0
C---
C---  *IF(THIS IS NOT A SPECIAL DISTRIBUTION TO BE HANDLED SEPERATELY)
       IF((PBINOM.LE.0.0) .OR. (PBINOM.GE. 1.0)
&                            .OR. (NTRYS.LE.4)) GO TO 3000
C---
C---     *DRAW RANDOM SAMPLE FROM UNIFORM (0,1) DISTRIBUTION
          RDRAW = UNIFM1(SEED)
C---
C---     *DETERMINE WHETHER TO SAMPLE SUCCESSES OR FAILURES
          PFAIL = AMAX1(PBINOM,1.0-PBINOM)
          PSUCC = 1.0 - PFAIL
C---
C---     *COMPUTE QUICK APPROXIMATION TO PROB(0 SUCCESSES)
          PROB = 1.0 - FLOAT(NTRYS)*PSUCC
          IF(RDRAW.LE.PROB) GO TO 2000
C---
C---     *COMPUTE EXACT PROBABILITY OF NO SUCCESSES
          PROB = PFAIL**NTRYS
C---
C---     *IF(RANDOM DRAW DOES NOT FALL WITHIN THIS PORTION OF THE CDF)
          IF(RDRAW.LE.PROB) GO TO 2000
C---
C---        *INITIALIZE LOOP TO FIND APPROPRIATE PLACE IN THE CDF
             RATIO = PSUCC/PFAIL
             NPLUS1 = NTRYS + 1
             CDF = PROB
C--
            *DO UNTIL(APPROPRIATE CDF INDEX IS FOUND)
 1000        CONTINUE
C--
C--            *UPDATE SAMPLE COUNTER
                NBINOM = NBINOM + 1
C--
C--            *COMPUTE NEXT ENTRY IN CUMULATIVE DISTRIBUTION FUNCTION
                PROB = (FLOAT(NPLUS1-NBINOM)/FLOAT(NBINOM))*RATIO*PROB
                CDF = CDF + PROB
C--
C--         *END DO (CDF LOOP)
             IF((RDRAW.GT.CDF) .AND. (NBINOM.LT.NTRYS)) GO TO 1000
C--
C--      *END IF (0 SUCCESSES TEST)
 2000     CONTINUE
C--
C--      *COMPLEMENT RESULT IF FAILURES WERE SAMPLED
          IF(PBINOM.GT. .5) NBINOM = NTRYS - NBINOM
C--
C--   *ELSE (SPECIAL CASES)
       GO TO 7000
 3000     CONTINUE
C--
C--      *IF(THIS IS A DEGENRATIVE DISTRIBUTION)THEN
          IF((PBINOM.GT.0).AND.(PBINOM.LT.1.0).AND.(NTRYS.GT.0))
&                                                GO TO 4000
C--
C--         *SAMPLE FROM DISTRIBUTION (IF PBINOM=0, OR NTRYS=0
C--                                     THEN WE ARE DONE)
             IF(PBINOM.GE.1.0) NBINOM = NTRYS
C--
C--      *ELSE (USE BERNOULLI TRIAL METHOD)
          GO TO 6000
 4000     CONTINUE
C--
C--         *PERFORM APPROPRIATE NUMBER OF BERNOULLI TRIALS
             DO 5000 I=1,NTRYS
                IF(UNIFM1(SEED).LE.PBINOM) NBINOM = NBINOM + 1
 5000        CONTINUE
C--
C--      *END IF (DEGENERATIVE DISTRIBUTION TEST)
 6000     CONTINUE
C--
C--   *END IF (SPECIAL CASES TEST)
 7000     CONTINUE
C--
    RETURN
    END
```

```
C****************************************************************
   INTEGER FUNCTION NDMNDS(NBRKAC)
C****************************************************************
C** NDMNDS     - GENERATE SAMPLE OF TOTAL SORTIE PART DEMANDS.
C***      NDMNDS IS A FORTRAN FUNCTION WHICH GENERATES A
C*** SAMPLE NUMBER OF PARTS DEMANDS ON A SORTIE, GIVEN THE TOTAL
C*** NUMBER OF AIRCRAFT WHICH BROKE ON THAT SORTIE. THE
C*** PROBABILITY DISTRIBUTION OF TOTAL PARTS DEMANDS IS APPROXIMATED
C*** USING EITHER A NORMAL DISTRIBUTION (IF MEAN IS LARGE ENOUGH TO
C*** APPLY THE CENTRAL LIMIT THEOREM) OR A POISSON DISTRIBUTION.
C***
C*** INPUTS --
C***   NBRKAC     - NUMBER OF AIRCRAFT WHICH BROKE ON THE SORTIE
C*** COMMON INPUTS --
C***   ACMEAN     - EXPECTED VALUE OF THE RANDOM VARIABLE REPRESENTING
C***                THE NUMBER OF PARTS DEMANDS PER AIRCRAFT, GIVEN
C***                THAT THE AIRCRAFT HAS BROKEN UPON RETURNING FROM
C***                A SORTIE.
C***   ACVAR      - VARIANCE OF TOTAL PARTS DEMAND PER BROKEN AIRCRAFT
C***   NPERAC     - TOTAL NUMBER OF PARTS PER AIRCRAFT. THIS IS USED TO
C***                ENSURE THAT A LEGITIMATE SAMPLE IS GENERATED.
C****************************************************************
C---
      PARAMETER CUTOFF=0.0
      COMMON /RSEED/  SEED
      COMMON /DEMAND/ ACMEAN, ACVAR, NPERAC
C---
C---      *INITIALIZE SAMPLE TO NO PARTS DEMANDS
           NDMNDS = 0
C---
C---      *IF(THERE WERE ANY BROKEN AIRCRAFT)THEN
           IF(NBRKAC.EQ.0) GO TO 300
C---
C---         *COMPUTE MEAN OF DISTRIBUTION OF TOTAL DEMANDS
C---          CORRESPONDING TO NUMBER OF BROKEN AIRCRAFT
              FLTAC  = FLOAT(NBRKAC)
              BMEAN  = FLTAC * ACMEAN
C---
C---         *IF(EXPECTED TOTAL DEMANDS IS SMALL)THEN
              IF(BMEAN.GT.CUTOFF) GO TO 100
C---
C---            *USE POISSON APPROXIMATION
                 NDMNDS = IPOISSON(BMEAN,SEED)
C---
C---         *ELSE
              GO TO 200
 100          CONTINUE
C---
C---            *USE NORMAL APPROXIMATION
                 BSTDEV = SQRT(FLTAC*ACVAR)
                 NDMNDS = MAX0(0,INT(XNORM(BMEAN,BSTDEV,SEED)+.5))
C---
C---         *END IF (APPROXIMATION TYPE TEST)
```

```
  200         CONTINUE
C---
C---         *ENSURE THAT A FEASIBLE ANSWER HAS BEEN GENERATED
              NDMNDS = MIN0(NDMNDS,NPERAC*NBRKAC)
C---
C---      *END IF (ZERO BROKEN AC TEST)
  300      CONTINUE
C---
   RETURN
   END
```

```
C*************************************************************
   INTEGER FUNCTION NORSAC(NPARTS,IQPA,NBACKO)
C*************************************************************
C** NORSAC     - CALCULATE INITIAL NUMBER OF NORS AIRCRAFT.
C***      NORSAC IS A FORTRAN FUNCTION WHICH CALCULATES THE CURRENT
C*** NUMBER OF NORS AIRCRAFT - ASSUMING PERFECT CANNIBALIZATION.
C***
C*** INPUT -
C***   NPARTS     - TOTAL NUMBER OF PART TYPES.
C***   IQPA(K)    - NUMBER OF TYPE-K PARTS INSTALLED ON EACH AIRCRAFT.
C***   NBACKO(K)  - NUMBER OF BACKORDERS FOR PARTS OF TYPE-K.IF
C***                NBACKO(K) IS POSITIVE, THEN UNFULFILLED REQUESTS
C***                FOR PARTS OF THIS TYPE HAVE BEEN MADE. IF IT IS
C***                NEGATIVE, THEN NBACKO(K) INDICATES THE NUMBER OF
C***                OF PARTS ON-THE-SHELF.
C*** OUTPUT -
C***   NORSAC     - CURRENT NUMBER OF NORS AIRCRAFT BASED ON THE GIVEN
C***                BACKORDER AND QPA INFORMATION AND ASSUMING PERFECT
C***                CANNABILIZATION.
C*************************************************************
C---
      DIMENSION NBACKO(NPARTS), IQPA(NPARTS)
C---
C---       *INITIALIZE NUMBER OF NORS AIRCRAFT TO NONE
           NORSAC = 0
C---
C---       *DO FOR(EACH PART TYPE)
           DO 2000 K=1,NPARTS
C---
C---          *IF(THESE PARTS CAUSE THE MAX NUMBER OF NORS THUS FAR)
              IF(NORSAC*IQPA(K) .GE. NBACKO(K)) GO TO 1000
C---
C---             *UPDATE NUMBER OF NORS AIRCRAFT
                  NORSAC=INT(FLOAT(NBACKO(K))/FLOAT(IQPA(K)) + .999)
C---
C---          *END IF (NEW NORS MAXIMUM TEST)
 1000          CONTINUE
C---
C---       *END DO (PARTS LOOP)
 2000      CONTINUE
C---
   RETURN
   END
```

```
C*****************************************************************
      INTEGER FUNCTION NORSBK(NBRKAC,NOROLD)
C*****************************************************************
C** NORSBK     - DETERMINES NORS AIRCRAFT FROM A SORTIE.
C***      NORSBK IS A FORTRAN FUNCTION WHICH CALCULATES THE NUMBER
C*** OF NORS AIRCRAFT RESULTING FROM A SORTIE WITH A SPECIFIED NUMBER
C*** OF BROKEN AIRCRAFT -- ASSUMING IMMEDIATE AND MAXIMUM
C*** CANNABILIZATION OF PARTS. NORSBK DETERMINES THE TOTAL
C*** NUMBER AND DISTRIBUTION OF THE PARTS DEMANDS RESULTING FROM THIS
C*** SORTIE. IT UPDATES FOR EACH PART TYPE DEMANDED, THE
C*** NUMBER OF PARTS ON-THE-SHELF, BACKORDERED, AND IN RESUPPLY.
C***
C*** INPUTS --
C***   NBRKAC     - NUMBER OF AIRCRAFT WHICH BROKE DURING THE SORTIE
C***   NOROLD     - NUMBER OF NORS AIRCRAFT BEFORE THIS LATEST SORTIE.
C*** COMMON INPUTS --
C***   IQPA(K)    - NUMBER OF TYPE-K PARTS INSTALLED ON EACH AIRCRAFT.
C***   INFPART    - LOGICAL FLAG INDICATING WHETHER THE INFINITE PARTS
C***                ASSUMPTION HOLDS. IF INFPART IS TRUE THEN THERE
C***                IS NEVER ANY SHORTAGE OF PARTS; HENCE, NO NORS AC.
C*** COMMON INPUTS/OUTPUTS -
C***   NBACKO(K)  - NUMBER OF BACKORDERS FOR PARTS OF TYPE-K.IF
C***                NBACKO(K) IS POSITIVE, THEN UNFULFILLED REQUESTS
C***                FOR PARTS OF THIS TYPE HAVE BEEN MADE. IF IT IS
C***                NEGATIVE, THEN NBACKO(K) INDICATES THE NUMBER
C***                OF PARTS ON-THE-SHELF.
C*** OUTPUT -
C***   NORSBK      - NUMBER OF NORS AIRCRAFT AT THE END OF THIS SORTIE
C***                 ASSUMING MAXIMUM AND IMMEDIATE CANNABILIZATION.
C*****************************************************************
C---
      PARAMETER MAXPRT=304, MAXCYC=10
      COMMON /PARTS/ NPARTS,IQPA(MAXPRT),NBACKO(MAXPRT),
     &   BRPRATE(MAXPRT),DRPRATE(MAXPRT),INITSJ(MAXPRT),RESUPP(MAXPRT),
     &   BNRTS(MAXPRT),NBASE(MAXPRT),NDEPOT(MAXPRT)
      COMMON /INPUT/   INITUE, NAC, PATTRIT, IRES, RNMCM, INFPART,
     &                 MAXFLY(MAXCYC), INFMAN, ISCALE, IAUGMNT
      LOGICAL INFPART
C---
C---       *INITIALIZE NEW NORS TO OLD NUMBER OF NORS AIRCRAFT
            NORSBK = NOROLD
C---
C---       *IF(THERE ARE ANY BROKEN AIRCRAFT AND
C---           INFINITE PARTS NOT ASSUMED)THEN
            IF(NBRKAC.EQ.0) GO TO 5000
            IF(INFPART) GO TO 5000
C---
C---          *DETERMINE TOTAL NUMBER OF PARTS DEMANDS FROM THESE BROKEN AC
               NDEMS = NDMNDS(NBRKAC)
C---
C---          *IF(ANY PARTS WERE DEMANDED)THEN
               IF(NDEMS.EQ.0) GO TO 4000
C---
```

```
C---              *DO FOR(EACH PART DEMAND)
                   DO 3000 I=1,NDEMS
C---
C---                 *DETERMINE PART-TYPE FOR THIS DEMAND BY SAMPLING
C---                  FROM A MULTNOMIAL DISTRIBUTION
                      KTYPE = MNOM()
C---
C---                 *UPDATE BACKORDERS FOR THIS PART TYPE
                      NBACKO(KTYPE) = NBACKO(KTYPE) + 1
C---
C---                 *IF(AN UNFULFILLED DEMAND HAS OCCURRED)THEN
                      IF(NBACKO(KTYPE).LE.0) GO TO 2000
C---
C---                    *IF(THIS DEMAND CAUSES A NEW NORS AC)THEN
                         IF(NORSBK*IQPA(KTYPE).GE.NBACKO(KTYPE))
&                          GO TO 1000
C---
C---                       *INCREMENT NUMBER OF NORS AIRCRAFT
                            NORSBK = NORSBK + 1
                           IF((NORSBK-NOROLD).GE.NBRKAC) GOTO 5000
C---
C---                    *END IF (NEW NORS AIRCRAFT TEST)
 1000                    CONTINUE
C---
C---                 *END IF (UNFULFILLED DEMAND TEST)
 2000                 CONTINUE
C---
C---              *END DO (DEMAND LOOP)
 3000              CONTINUE
C---
C---           *END IF (ZERO DEMANDS TEST)
 4000           CONTINUE
C---
C---         *END IF (NO BROKEN AC OR INFINITE PARTS TEST)
 5000         CONTINUE
C---
   RETURN
   END
```

```
C*****************************************************************
   INTEGER FUNCTION NREPS(TIMET,NREPJ,NCRWSJ,SRATEJ)
C*****************************************************************
C** NREPS      - RANDOM SAMPLE OF AIRCRAFT REPAIRS IN A WORK CENTER.
C***     NREPS IS A FORTRAN FUNCTION WHICH RETURNS A SAMPLE NUMBER
C*** OF AIRCRAFT REPAIRED IN A WORKCENTER, BASED ON THE LENGTH OF THE
C*** REPAIR PERIOD, NUMBER OF SERVERS, REPAIR RATE FOR EACH SERVER,
C*** AND THE NUMBER OF AIRCRAFT IN THE WORKCENTER AT THE START OF
C*** THE REPAIR PERIOD. IT IS ASSUMED THAT NO NEW AIRCRAFT ARRIVE
C*** DURING THE REPAIR PERIOD, AND REPAIR TIMES ARE EXPONENTIALLY
C*** DISTRIBUTED, WITH THE SAME DISTRIBUTION APPYING TO EACH SERVER
C*** INDEPENDENTLY.
C***
C*** INPUTS -
C***   TIMET      - LENGTH (IN HOURS) OF THE REPAIR PERIOD.
C***   NREPJ      - NUMBER OF AIRCRAFT IN THE WORKCENTER AT THE
C***                START OF THE REPAIR PERIOD.
C***   NCRWSJ     - NUMBER OF REPAIR CREWS (SERVERS) FOR THIS
C***                WORKCENTER.
C***   SRATEJ     - REPAIR RATE (AIRCRAFT/HOUR) FOR EACH CREW IN
C***                THIS WORKCENTER.
C*** COMMON INPUT/OUTPUT --
C***   SEED       - SEED FOR RANDOM NUMBER GENERATOR.
C*** OUTPUT -
C***   NREPS      - NUMBER OF AIRCRAFT REPAIRED IN THIS WORKCENTER
C***                DURING THE REPAIR PERIOD.  NREPS=0,1,2,...,NREPJ
C*****************************************************************
C---
      COMMON /RSEED/SEED
C---
C---        *IF(NUMBER OF AC IN REPAIR IS LESS THAN THE NUMBER OF CREWS)
             IF(NREPJ.GT.NCRWSJ) GO TO 1000
C---
C---           *DETERMINE NUMBER OF AIRCRAFT REPAIRED BY SAMPLING
C---                        FROM THE APPROPRIATE BINOMIAL DISTRIBUTION
                NREPS = NREPJ - NBINOM( EXP(-SRATEJ*TIMET) , NREPJ )
C---
C---        *ELSE (MORE AIRCRAFT THAN CREWS)
             GO TO 4000
 1000        CONTINUE
C---
C---           *INITIALIZE VARIABLES
                NREPS  = 0
                CUMP   = 1.0
                MAXREP = NREPJ - NCRWSJ + 1
                CWRATE = FLOAT(NCRWSJ)*SRATEJ
                EXPTYM = EXP(-CWRATE*TIMET)
C---
C---           *DO UNTIL(A SERVER BECOMES IDLE OR THE NEXT AIRCRAFT
C---                    DEPARTURE TIME EXCEEDS LENGTH OF REPAIR PERIOD)
 2000           CONTINUE
C---
C---              *GENERATE AND ACCUMULATE NEXT AIRCRAFT DEPARTURE FROM
```

```
C---                                          THIS WORKCENTER
                  CUMP = CUMP * UNIFM1(SEED)
C---
C---             *EXIT LOOP, IF REPAIR TIMES EXCEED TIME INTERVAL LENGTH
                  IF(CUMP .LT. EXPTYM) GO TO 3000
C---
C---             *INCREMENT NUMBER OF AIRCRAFT REPAIRED
                  NREPS = NREPS + 1
C---
C---          *END DO (REPAIRED AIRCRAFT LOOP)
               IF(NREPS.LT.MAXREP) GO TO 2000
C---
C---          **A SERVER HAS JUST BECOME IDLE, PERFORM A BINOMIAL DRAW
C---            TO DETERMINE HOW MANY MORE AC ARE REPAIRED
C---
C---             *COMPUTE TIME LEFT IN THE INTERVAL
C---                (LENGTH OF REPAIR PERIOD) - (TIME OF LAST REPAIR)
                  TLEFT = TIMET + ALOG(CUMP)/CWRATE
C---
C---             *COMPUTE PROBABILITY AN AIRCRAFT IS NOT REPAIRED
C---              IN THE REMAINDER OF THE INTERVAL
                  PNOREP = EXP(-SRATEJ*TLEFT)
C---
C---             *GENERATE A BINOMIAL DRAW TO DETERMINE NUMBER OF
C---                         REMAINING AIRCRAFT WHICH ARE NOT REPAIRED
                  NOTREP=NBINOM(PNOREP,NCRWSJ-1)
C---
C---             *COMPUTE TOTAL AIRCRAFT REPAIRED DURING PERIOD
                  NREPS=NREPJ-NOTREP
C---
C---
C---          *EXIT FROM DO LOOP
 3000          CONTINUE
C---
C---       *END IF (MORE AIRCRAFT THAN CREWS TEST)
 4000       CONTINUE
C---
   RETURN
   END
```

```
C****************************************************************
   SUWROUTINE PRINTO
C****************************************************************
C** PRINTO     - PRINT-OUT RESULTS OF THE SIMULATION RUN.
C***      THIS ROUTINE PRINTS THE RESULTS OF THE SGM SIMULATION.
C*** THESE RESULTS CONSIST OF THE AVERAGE NUMBERS OF AIRCRAFT IN
C*** THE VARIOUS POSSIBLE AIRCRAFT STATES AT THE START OF EACH
C*** SORTIE PERIOD FOR EACH FLYING DAY OF THE SCENARIO. THE AVERAGE
C*** SORTIES PER AIRCRAFT PER DAY IS COMPUTED AND ALSO PRINTED.
C*** THESE SORTIES PER AIRCRAFT PER DAY FIGURES AND THE TOTAL SORTIE
C*** PRODUCTION PER DAY ARE PRINTED TO A SCRATCH FILE (FILE 07) TO BE
C*** USED AS INPUT FOR SORTIE PLOTS.
C****************************************************************
C---
      PARAMETER  MAXAC=108,MAXWC=25,MAXBIT=36,MAXPRT=304,
&                  MAXVEC=2+(MAXAC-1)/MAXBIT
      PARAMETER MAXDAY=30,MAXCYC=10,MAXSTAT=5
      COMMON /STATS/   EXPECT(MAXSTAT,MAXCYC,MAXDAY),
&                      NRESRV, IZDAY,ITOTRES(MAXDAY), LOSSTOT
      COMMON /TIME/ PREFLITE,SORTLGTH,WAITCYC,
&       TYMNITE,NSIM,ISIM,NUMDAY,IDAY,NCYCLES,ICYCLE
      COMMON /INPUT/  INITUE,NAC,PATTRIT,IRES,RNMCM,INFPART,
&          MAXFLY(MAXCYC),INFMAN,ISCALE,IAUGMNT
      COMMON /WCINPUT/ NWC, NCREWS(MAXWC), SRATE(MAXWC)
C---
C---  *PRINT EXPECTED NUMBER OF AVAILABLE AIRCRAFT FOR EACH SORTIE PERIOD
       WRITE (6,9005)
       TOTFLY = 0.0
       WRITE (7) FLOAT(ISCALE),NUMDAY
       FSIM = FLOAT(NSIM)
       DO 6000 J=1,NUMDAY
          SORTYDAY = EXPECT(1,1,J)
          OFFSCENE = EXPECT(4,1,J)+EXPECT(5,1,J)
          WRITE(6,9008) J,1,(EXPECT(M,1,J)/FSIM,M=1,5)
          DO 5000 I=2,NCYCLES-1
             WRITE (6,9006) I,(EXPECT(M,I,J)/FSIM,M=1,4)
             SORTYDAY = SORTYDAY + EXPECT(1,I,J)
             OFFSCENE = OFFSCENE + EXPECT(4,I,J) + EXPECT(5,I,J)
 5000     CONTINUE
          I = NCYCLES
          SORTYDAY = (SORTYDAY + EXPECT(1,I,J))/FSIM
          OFFSCENE = (OFFSCENE + EXPECT(4,I,J) + EXPECT(5,I,J))/FSIM
          AONSCENE = NCYCLES*(INITUE+ITOTRES(J)) - OFFSCENE
          SORTYAC=NCYCLES*SORTYDAY/AONSCENE
          WRITE (6,9009) I,EXPECT(1,I,J)/FSIM,SORTYDAY,
&            SORTYAC,(EXPECT(M,I,J)/FSIM,M=2,4)
          TOTFLY = TOTFLY + SORTYDAY
C---
C---     *WRITE THE MEAN SORTIES PER DAY (FOR EACH DAY) TO A FILE THAT
C---         COULD BE USED BY 'CALLPLT2' TO PRODUCE A PLOT.
          WRITE (7) SORTYAC, SORTYDAY
          WRITE(6,9003) TOTFLY
 6000  CONTINUE
```

```
      WRITE (6,9010) TOTFLY
C---
   RETURN
 9002  FORMAT(V)
 9003  FORMAT (F26.1)
 9005  FORMAT ('1'///10X,'SORTIES/  SORTIES/  SORTIES/',21X,
&          'CUM.   RES.'/
&          ' DAY  PER  PERIOD',5X,'DAY',8X,'AC',4X,
&              '   NMCM     NMCS   LOSSES  LEFT'//)
 9006  FORMAT (' ',2X,I5,F9.1,F29.1,F9.1,F8.1)
 9008  FORMAT (' ',I2,I5,F9.1,F29.1,F9.1,F8.1,F8.1)
 9009  FORMAT (' ',2X,I5,F9.1,F9.1,F11.2,2F9.1,2F8.1)
 9010  FORMAT (///' TOTAL SORTIES FLOWN =',F11.1)
   END
```

```
C*****************************************************************
   SUBROUTINE PRTREP(RTIME,PBRKSEQ,INDXWC,NORSVC)
C*****************************************************************
C** PRTREP     - SIMULATES PROCESS OF REPAIRING PARTS.
C***       PRTREP IS A FORTRAN SUBROUTINE WHICH SIMULATES THE PROCESS
C*** OF REPAIRING PARTS. PARTS REPAIR IS ASSUMED TO BE
C*** EXPONENTIAL WITH AN INFINITE NUMBER OF SERVERS, I.E. NO PART
C*** EVER HAS TO WAIT TO BEGIN SERVICE. IN AUDITION TO REPAIRING
C*** PARTS, THIS FUNCTION CALCULATES THE NEW NUMBER OF NORS AIRCRAFT
C*** REMAINING AFTER REPAIR OF THESE PARTS. IF ANY PREVIOUSLY NORS
C*** AIRCRAFT ARE READY TO GO INTO MAINTENANCE, THIS ROUTINE
C*** WILL DISTRIBUTE THEM PROBABILISTICALLY AMONG THE VARIOUS
C*** WORKCENTERS.
C***
C*** INPUT -
C***   RTIME      - AVERAGE TIME (IN HOURS) THESE PARTS HAVE BEEN
C***                IN REPAIR SINCE THE LAST TIME PARTS REPAIR WAS
C***                SIMULATED.
C***   PBRKSEQ    - 2-DIMENSIONAL ARRAY USED TO DETERMINE THE
C***                DISTRIBUTION OF ABORTS INTO THE VARIOUS
C***                WORKCENTERS.
C*** INPUT/OUTPUT -
C***   NORSVC     - NORS AIRCRAFT STATUS VECTOR. INDICATES THOSE
C***                AIRCRAFT WHICH ARE NORS DUE TO UNAVAILABLE PARTS.
C***                THE FIRST WORD, NORSVC(1), CONTAINS THE TOTAL
C***                NUMBER OF 1-BITS IN THE NORS STATUS VECTOR.
C***                ARRAY IS A BIT VECTOR WITH EACH BIT REPRESENTING
C***                AN AIRCRAFT. A 1-BIT INDICATES THE AIRCRAFT IS
C***                STILL FLYABLE. NOTE THAT IFLYVC(1) ALSO INDICATES
C***                THE NUMBER OF 1-BITS IN THIS BIT VECTOR.
C*** COMMON INPUT -
C***   INFPART    - LOGICAL FLAG INDICATING WHETHER THE INFINITE PARTS
C***                ASSUMPTION HOLDS. IF INFPART IS TRUE THEN THERE
C***                IS NEVER ANY SHORTAGE OF PARTS; HENCE, NO
C***                NORS AC.
C***   NPARTS     - NUMBER OF PART TYPES BEING MODELED.
C***   IQPA(K)    - NUMBER OF TYPE-K PARTS INSTALLED ON EACH AIRCRAFT.
C***   RPRATE(K)  - REPAIR RATE (PARTS/HOUR) FOR TYPE-K PARTS
C***   INITSJ(K)  - INITIAL BASE STOCK LEVEL FOR TYPE-K PARTS.
C*** COMMON INPUT/OUTPUT -
C***   NBACKO(K)  - NUMBER OF BACKORDERS FOR PARTS OF TYPE-K.IF
C***                NBACKO(K) IS POSITIVE, THEN UNFULFILLED REQUESTS
C***                FOR PARTS OF THIS TYPE HAVE BEEN MADE. IF IT IS
C***                NEGATIVE, THEN NBACKO(K) INDICATES THE NUMBER OF
C***                OF PARTS ON-THE-SHELF.
C*****************************************************************
C---
      PARAMETER MAXPRT=304, MAXCYC=10
      COMMON /PARTS/ NPARTS,IQPA(MAXPRT),NBACKO(MAXPRT),
&        BRPRATE(MAXPRT),DRPRATE(MAXPRT),INITSJ(MAXPRT),RESUPP(MAXPRT),
&        BNRTS(MAXPRT),NBASE(MAXPRT),NDEPOT(MAXPRT)
      COMMON /INPUT/   INITUE, NAC, PATTRIT, IRES, RNMCM, INFPART,
&                      MAXFLY(MAXCYC), INFMAN, ISCALE, IAUGMNT
```

```
      DIMENSION NORSVC(1)
      LOGICAL INFPART
C--
C--     *INITIALIZE NEW NUMBER OF NORS AIRCRAFT TO NONE
         NEWNOR = 0
C--
C--     *IF(INFINITE PARTS NOT ASSUMED)THEN
         IF(INFPART) GO TO 5000
C--
C--       *DO FOR(EACH PART TYPE)
           DO 3000 K=1,NPARTS
C--
C--         *IF(THERE ARE ANY OF THIS PART IN REPAIR)THEN
             INSHPK = NBACKO(K) + INITSJ(K)
             IF(INSHPK.LE.0) GO TO 2000
C--            *DETERMINE NUMBER OF THESE WHICH ARE NEW DEMANDS
                NEW = INSHPK - (NDEPOT(K) + NBASE(K))
C--
C--            *PERFORM BINOMIAL DRAW TO DETERMINE BASE/DEPOT SPLIT
                NEWDEP=NBINOM(BNRTS(K),NEW)
                NDEPOT(K)=NDEPOT(K)+NEWDEP
                NBASE(K)=NBASE(K)+(NEW-NEWDEP)
C--
C--            *COMPUTE PROBABILITY OF REPAIR
                PDEP = 1.0 - EXP(-DRPRATE(K)*RTIME)
                PBSE = 1.0 - EXP(-BRPRATE(K)*RTIME)
C--
C--            *DETERMINE NUMBER OF PARTS REPAIRED BY SAMPLING
C--             FROM THE APPROPRIATE BINOMIAL DISTRIBUTION
                NUMDEP = NBINOM(PDEP,NDEPOT(K))
                NUMBSE = NBINOM(PBSE,NBASE(K))
C--
C--            *UPDATE NUMBER IN-SHOP AND BACKORDERED
                NBACKO(K) = NBACKO(K) - (NUMDEP + NUMBSE)
                NDEPOT(K) = NDEPOT(K) - NUMDEP
                NBASE(K) = NBASE(K) - NUMBSE
C--
C--            *IF(THESE PARTS CAUSE THE MAX NUMBER OF NORS THUS FAR)
                IF(NEWNOR*IQPA(K) .GE. NBACKO(K)) GO TO 1000
C--
C--              *UPDATE NUMBER OF NORS AIRCRAFT
                  NEWNOR=INT(FLOAT(NBACKO(K))/FLOAT(IQPA(K)) + .999)
C--
C--            *END IF (NEW NORS MAXIMUM TEST)
 1000           CONTINUE
C--
C--         *END IF (NO PARTS IN REPAIR TEST)
 2000        CONTINUE
C--
C--       *END DO (PARTS LOOP)
 3000      CONTINUE
C--
C--       *IF(ANY PREVIOUSLY NORS AIRCRAFT ARE READY FOR REPAIR)THEN
```

```
             NORDIF = NORSVC(1) - NEWNOR
             IF(NORDIF.LE.0) GO TO 4000
C--
C--            *SELECT LEFTMOST NORS AIRCRAFT TO ENTER MAINTENANCE
                CALL WCDIST(NORDIF,PBRKSEQ,INDXWC,NORSVC)
C--
C--          *END IF (NEW NONNORS AC TEST)
 4000        CONTINUE
C--
C--      *END IF (INFINITE PARTS TEST)
 5000     CONTINUE
C--
   RETURN
   END
```

```
C****************************************************************
   SUBROUTINE PSTAT(PBREAK,NPARTS,IQPA,DEMAND,
&                                     ACMEAN,ACVAR,NPERAC)
C****************************************************************
C** PSTAT      - CALCULATES STATISTICS FOR TOTAL PART DEMANDS.
C***       PSTAT IS A FORTRAN ROUTINE WHICH CALCULATES THE MEAN
C*** AND VARIANCE OF THE RANDOM VARIABLE REPRESENTING THE TOTAL
C*** NUMBER OF PART DEMANDS FROM AN AIRCRAFT WHICH HAS BROKEN UPON
C*** RETURNING FROM A SORTIE.
C***
C*** INPUTS -
C***   PBREAK     - PROBABILITY THAT AN AIRCRAFT BREAKS UPON RETURNING
C***                FROM A SORTIE.
C***   NPARTS     - NUMBER OF PART TYPES BEING MODELED FOR THIS TYPE
C***                OF AIRCRAFT.
C***   IQPA(K)    - QUANTITY PER AIRCRAFT FOR TYPE-K PARTS.
C***   DEMAND(K)  - PROBABILITY THAT A GIVEN TYPE-K PART WILL BE
C***                DEMANDED BY AN AIRCRAFT RETURNING FROM A SORTIE.
C***  OUTPUTS -
C***   ACMEAN     - MEAN OF THE RANDOM VARIABLE REPRESENTING NUMBER
C***                OF PART DEMANDS PER BROKEN AIRCRAFT.
C***   ACVAR      - VARIANCE OF TOTAL PART DEMANDS PER BROKEN AIRCRAFT.
C***   NPERAC     - TOTAL NUMBER OF PARTS PER AIRCRAFT. THIS IS USED
C***                TO ENSURE THAT A LEGITIMATE SAMPLE IS GENERATED.
C****************************************************************
C---
     DIMENSION IQPA(NPARTS), DEMAND(NPARTS)
C---
C---  *INITIALIZE STATISTICS
      ACMEAN = 0.0
      ACVAR  = 0.0
      NPERAC = 0
C---
C---  *DO FOR(EACH PART TYPE)
      DO 1000  K=1,NPARTS
C---
C---     *ACCUMULATE STATISTICS
         PRTYPE = IQPA(K) * DEMAND(K)
         ACMEAN = ACMEAN + PRTYPE
         ACVAR  = ACVAR + PRTYPE*(PBREAK-DEMAND(K))
         NPERAC = NPERAC + IQPA(K)
C---
C---  *END DO (PART LOOP)
 1000  CONTINUE
C---
C---  *COMPLETE MEAN/VARIANCE COMPUTATIONS
      ACMEAN = ACMEAN/PBREAK
      ACVAR  = ACVAR/(PBREAK*PBREAK)
C---
   RETURN
   END
```

```
C***********************************************************************
   SUBROUTINE REPAIR(TIMET,NWC,NCREWS,SRATE)
C***********************************************************************
C** REPAIR     - SIMULATES PROCESS OF WORK CENTER AIRCRAFT REPAIR.
C***      REPAIR IS A FORTRAN ROUTINE WHICH SIMULATES THE
C*** PROCESS OF REPAIRING AIRCRAFT IN ALL WORKCENTERS DURING A
C*** SPECIFIED TIME PERIOD. A NUMBER OF REPAIRED AIRCRAFT IS GENERATED
C*** FOR EACH WORKCENTER, BY SAMPLING FROM THE APPROPRIATE PROBABILITY
C*** DISTRIBUTIONS. THIS ROUTINE SIMPLY UPDATES THE TOTAL NO. OF
C*** AIRCRAFT IN REPAIR IN EACH WORKCENTER; IT DOES NOT CONCERN
C*** ITSELF WITH WHICH PARTICULAR AIRCRAFT IN A W/C ARE
C*** BEING REPAIRED, NOR THE TOTAL NO. OF AIRCRAFT IN MAINTENANCE.
C*** IMPLICITLY, IT IS ASSUMED THAT WE ARE REPAIRING THE RIGHTMOST
C*** AIRCRAFT ON A LIST OF AIRCRAFT THAT NEED WORK IN A GIVEN W/C,
C*** AND THAT IF AIRCRAFT ARE PLACED ON THAT LIST (IN ROUTINE
C*** 'WCDIST') IN A RANDOM ORDER, THEN THIS METHOD OF REPAIR IS
C*** ALSO RANDOM; I.E., IT DOESN'T FAVOR LOW-NUMBERED A/C, OR
C*** HIGH-NUMBERED A/C, OR RECENTLY-BROKEN A/C, ETC.
C***
C*** INPUT --
C***   TIMET      - LENGTH (IN HOURS) OF THE REPAIR PERIOD.
C***   NWC        - TOTAL NUMBER OF WORKCENTERS.
C***   NCREWS     - NUMBER OF CREWS IN WORKCENTER-J.
C***   SRATE(J)   - REPAIR RATE (AIRCRAFT/HOUR) FOR EACH CREW IN
C***                WORKCENTER-J.
C*** COMMON INPUTS/OUTPUTS --
C***   INREPR(J)  - NO. OF A/C IN MAINTENANCE IN W/C J.
C***********************************************************************
C--
      PARAMETER MAXWC=25,MAXBIT=36,MAXAC=108,MAXVEC=2+(MAXAC-1)/MAXBIT
      PARAMETER  MAXDAY=30
      PARAMETER LFLD=7,NPERWRD=MAXBIT/LFLD,MXINWC=1+(MAXAC-1)/NPERWRD
      DIMENSION NCREWS(NWC), SRATE(NWC)
      COMMON /WCMAINT/ LISTRP(MXINWC,MAXWC),INREPR(MAXWC)
C--
C--        *DO FOR(EACH WORKCENTER)
            DO 400  J=1,NWC
C--
C--           *IF(THERE ARE ANY AIRCRAFT IN REPAIR IN THIS WORKCENTER)
               NACINJ = INREPR(J)
               IF(NACINJ.EQ.0) GO TO  200
C--
C--              *GENERATE SAMPLE NUMBER OF AIRCRAFT REPAIRED
                  NFIXED = NREPS(TIMET,NACINJ,NCREWS(J),SRATE(J))
C--
C--              *UPDATE NO. OF A/C IN THIS W/C
                  INREPR(J) = NACINJ - NFIXED
C--
C--           *END IF (ZERO AIRCRAFT IN REPAIR TEST)
  200          CONTINUE
C--
C--        *END DO (WORKCENTER LOOP)
  400       CONTINUE
```

```
C---
      RETURN
      END
```

```
C*****************************************************************
   SUBROUTINE SIMULA
C*****************************************************************
C** SIMULA     - PERFORM SIMULATION REPLICATIONS.
C***     THIS ROUTINE PROVIDES THE BASIC STRUCTURE OF THE SGM
C*** SIMULATION. THE LOOPS WHICH CONTROL THE SIMULATION REPLICATIONS,
C*** THE FLYING DAYS FOR EACH REPLICATION, AND THE FLYING CYCLES
C*** FOR EACH FLYING DAY ARE CONTAINED IN THIS ROUTINE. THE
C*** ROUTINE BASICALLY JUST EXECUTES A SPECIFIED NUMBER OF
C*** FLYING CYCLES EACH FLYING DAY.
C***   THE ROUTINE READS A SUBSET OF SCENARIO PARAMETERS FOR EACH
C*** FLYING DAY OF THE SCENARIO FROM A SCRATCH FILE (03) WHICH
C*** WAS INITIALIZED IN THE INITSCN SUBROUTINE. THIS APPROACH ALLOWS
C*** THESE PARAMETERS TO VARY ON A DAILY BASIS DURING THE
C*** SIMULATION.
C***
C*** COMMON INPUTS --
C***   NSIM       - NUMBER OF SIMULATION REPLICATIONS TO BE PERFORMED
C***   NUMDAY     - NUMBER OF FLYING DAYS TO SIMULATE
C***   NCYCLES    - NUMBER OF FLYING CYCLES FOR EACH DAY. THIS PARAMETER
C***                IS READ FROM THE SCRATCH FILE EACH SIMULATION DAY
C*** COMMON OUTPUTS --
C***    ISIM      - (ISIM=1,...,NSIM) CURRENT REPLICATION NUMBER
C***    IDAY      - (IDAY=1,...,NUMDAY) CURRENT FLYING DAY
C***    ICYCLE    - (ICYCLE=1,...,NCYCLES) CURRENT FLYING CYCLE
C***    NRESRV    - CURRENT NUMBER OF AIRCRAFT IN THE RESERVE POOL
C***    ITOTRES(I)- (I=0,...,NUMDAY) CUMULATIVE NUMBER OF AVAILABLE
C***                RESERVE AIRCRAFT UP TO AND INCLUDING THE ITH DAY.
C***                THE 0TH DAY REPRESENTS THE INITIAL NUMBER OF RESERVE
C***                AIRCRAFT. THIS ARRAY IS USED IN COMPUTING ON-THE-
C***                SCENE AIRCRAFT FOR SORTIES/AIRCRAFT/DAY IN THE
C***                PRINTO ROUTINE.
C***    IRES      - NUMBER OF RESERVE AIRCRAFT WHICH BECOME AVAILABLE
C***                ON THIS FLYING DAY.
C***    PATTRIT,PACGABT,WAITCYC,TYMNITE,MAXFLY(I)
C***               - SCENARIO PARAMETERS WHICH ARE ALLOWED TO VARY
C***                ON A DAILY BASIS. THEY ARE LOADED IN THIS
C***                ROUTINE AND SUPPLIED TO THE FLYCYC ROUTINE
C*****************************************************************
C--
      PARAMETER MAXWC=25, MAXDAY=30, MAXCYC=10, MAXSTAT=5
      COMMON /INPUT/   INITUE, NAC, PATTRIT, IRES, RNMCM, INFPART,
&                      MAXFLY(MAXCYC), INFMAN, ISCALE, IAUGMNT
      COMMON /STATS/   EXPECT(MAXSTAT,MAXCYC,MAXDAY),
&                      NRESRV, IZDAY,ITOTRES(MAXDAY), LOSSTOT
      COMMON /TIME/    PREFLITE, SORTLGTH, WAITCYC, TYMNITE,
&                      NSIM, ISIM, NUMDAY, IDAY, NCYCLES, ICYCLE
      COMMON /WCBRK/   PACBRK, PACGABT, PBRKWC(MAXWC), PWCPROD,
&                      PBRKSEQ(2,MAXWC), INDXWC(MAXWC)
C--
C--  *DO FOR(EACH SIMULATION REALIZATION)
       DO 300 ISIM=1,NSIM
C--
```

```
C---    *INITIALIZE VARIABLES AT START OF EACH REPLICATION
         CALL INITREP
C---
C---    *REWIND DAILY SCENARIO PARAMETERS FILE
         REWIND 03
C---
C---    *DO FOR(EACH FLYING DAY)
         DO 200 IDAY=1,NUMDAY
C---
C---      *READ DAILY SCENARIO PARAMETERS
           READ(03) PATTRIT,PACGABT,NCYCLES,IRES,WAITCYC,TYMNITE
           READ(03) (MAXFLY(J),J=1,NCYCLES)
C---
C---      *UPDATE RESERVE AC POOL WITH NEW AVAILABLE RESERVES
           NRESRV = NRESRV + IRES
           ITOTRES(IDAY) = ITOTRES(IDAY-1) + IRES
C---
C---      *DO FOR(EACH FLYING CYCLE)
           DO 100 ICYCLE=1,NCYCLES
C---
C---        *SIMULATE A FLYING CYCLE
             CALL FLYCYC
C---
C---      *END DO (CYCLE LOOP)
  100      CONTINUE
C---
C---    *END DO (DAY LOOP)
  200    CONTINUE
C---
C--- *END DO (REPLICATION LOOP)
  300 CONTINUE
C---
   RETURN
   END
```

```
C****************************************************************
   SUBROUTINE SORTDS(NRECS,RKEYS,INDEX)
C****************************************************************
C** SORTDS     - DESCENDING SORT OF A REAL ARRAY.
C***      THIS ROUTINE RETURNS A SORTED LIST OF INDICES ACCORDING
C*** TO THE GIVEN ARRAY OF FLOATING-POINT KEYS.  THIS IS A
C*** DESCENDING SORT (I.E. LARGEST TO SMALLEST) USING THE
C*** SIMPLE EXCHANGE SORT TECHNIQUE.
C***
C*** INPUTS --
C***      NRECS    - NUMBER OF ENTRIES OR RECORDS IN THE INPUT AND
C***                 OUTPUT ARRAYS.
C***      RKEYS    - ARRAY OF FLOATING-POINT KEYS WHOSE CORRESPONDING
C***                 INDICES ARE TO BE SORTED.
C*** OUTPUTS --
C***      INDEX    - ARRAY OF INDICES, 1,...,NRECS , SORTED IN DESCENDING
C***                 ORDER ACCORDING TO THE CORRESPUNDING RKEY ENTRY.
C***                 THUS, RKEY( INDEX(1) ) IS THE LARGEST ENTRY, AND
C***                 RKEY( INDEX(NRECS) ) IS THE SMALLEST.
C****************************************************************
C--
      DIMENSION RKEYS(NRECS), INDEX(NRECS)
C--
C--   *INITIALIZE INDEX ARRAY TO NATURAL ORDER
       DO 100 I=1,NRECS
          INDEX(I)=I
  100  CONTINUE
C--
C--   *IF(THERE IS MORE THAN 1 RECORD)THEN
       IF(NRECS.LE.1)GO TO 500
C--
C--      *DO UNTIL(NO INTERCHANGES OCCUR)
          LIMIT=NRECS-1
  200     CONTINUE
C--
C--         *INITIALIZE INTERCHANGE FLAG TO NONE
             LASTX=0
C--
C--         *DO FOR(EACH CNSECUTIVE PAIR OF RECORDS)
             DO 400 J=1,LIMIT
C--
C--            *IF(INDICES OF THESE RECORDS SHOULD BE INTERCHANGED)
                IFIRST = INDEX(J)
                ISEC   = INDEX(J+1)
                IF(RKEYS(IFIRST).GE.RKEYS(ISEC)) GO TU 300
C--
C--               *INTERCHANGE INDICES AND MARK LAST EXCHANGE LOCATION
                   INDEX(J)   = ISEC
                   INDEX(J+1) = IFIRST
                   LASTX = J
C--
C--            *END IF (INTERCHANGE TEST)
  300           CONTINUE
```

```
C---
C---         *END DO (RECORD PAIR LOOP)
  400         CONTINUE
C---
C---      *END DO (INTERCHANGE PASS LOOP)
           LIMIT = LASTX
           IF(LASTX.NE.0) GO TO 200
C---
C---   *END IF (SINGLE RECORD TEST)
  500   CONTINUE
C---
   RETURN
   END
```

```
C*****************************************************************
      SUBROUTINE TBITSL(NONES,IFROM,ITO)
C*****************************************************************
C** TBITSL     - TRANSFER 1-BITS FROM LEFT OF A BIT-VECTOR.
C***       TBITSL IS A FORTRAN SUBROUTINE WHICH WILL TRANSFER
C*** A SPECIFIED NUMBER OF 1-BITS IN THE LEFTMOST PORTION OF
C*** A BIT-VECTOR, 'IFROM', TO THE CORRESPONDING POSITIONS
C*** OF A BIT-VECTOR, 'ITO'. THE TRANSFERRED BITS ARE THEN
C*** ZEROED OUT IN 'IFROM'. THESE BIT-VECTORS ARE KEPT IN
C*** ARRAYS, ORGANIZED IN THE FOLLOWING MANNER - THE FIRST WORD
C*** OF THE ARRAY CONTAINS THE CURRENT NUMBER OF 1-BITS IN THE
C*** BIT-VECTOR, AND THE REMAINING WORDS OF THE ARRAY CONTAIN THE
C*** ACTUAL BIT-VECTOR.
C***
C*** INPUT -
C***   NONES      - NUMBER OF 1-BITS TO BE TRANSFERRED. NOTE THAT
C***                NONES MUST NOT BE GREATER THAN THE NUMBER OF
C***                1S IN THE BIT STRING, 'IFROM.
C*** INPUT/OUTPUT -
C***   IFROM      - ARRAY CONTAINING THE BIT-VECTOR FROM WHICH
C***                THE LEFTMOST 1-BIT POSITIONS ARE TO BE
C***                TRANSFERRED. NOTE THAT THESE LEFTMOST
C***                1-BITS ARE ZEROED OUT IN 'IFROM' UPON
C***                COMPLETION OF THIS ROUTINE. IFROM(1) IS
C***                A COUNTER WHICH INDICATES THE CURRENT
C***                NUMBER OF 1-BITS IN THE BIT-VECTOR. THE
C***                ACTUAL BIT-VECTOR IS THE CONSECUTIVE BITS
C***                CONTAINED IN THE WORDS IFROM(2)-IFROM(LENGTH)
C***    ITO       - ARRAY CONTAINING THE BIT-VECTOR TO WHICH
C***                THE LEFTMOST 1-BITS IN 'IFROM' ARE TO BE
C***                TRANSFERRED. THE RIGHTMOST POSITIONS IN ITO
C***                REMAIN AS BEFORE. ITO(1) IS A COUNTER WHICH
C***                INDICATES THE CURRENT NUMBER OF 1-BITS IN
C***                THE BIT-VECTOR. THE ACTUAL BIT VECTOR IS THE
C***                CONSECUTIVE BITS CONTAINED IN THE WORDS -
C***                        ITO(2) - ITO(LENGTH)
C*** COMMON INPUT -
C***   LENGTH     - LENGTH (IN COMPUTER WORDS) OF THE ARRAYS
C***                CONTAINING THE VARIOUS BIT-VECTORS; IFLYVC, ETC
C*****************************************************************
C--
      PARAMETER MAXAC=108,MAXBIT=36,MAXVEC=2+(MAXAC-1)/MAXBIT
      COMMON /ACSTATE/ LENGTH,NACVC(MAXVEC), IFLYVC(MAXVEC),
     &                MAINVC(MAXVEC),NORSVC(MAXVEC), LOSTVC(MAXVEC)
      DIMENSION IFROM(1), ITO(1)
C--
C--       *INITIALIZE TRANSFER LOOP
           NLEFT  = NONES
           INDEX  = 2
C--
C--       *DO WHILE(ALL APPROPRIATE WORDS HAVE NOT BEEN MODIFIED)
 1000      CONTINUE
           IF(NLEFT.LE.0)       GO TO 4000
```

```
         IF(INDEX.GT.LENGTH) GO TO 4000
C--
C--        *COUNT NUMBER OF 1S IN NEXT WORD
            NXT1S  = N1BITS( IFROM(INDEX) )
C--
C--        *IF(NOT ALL 1-BITS IN THIS WORD SHOULD BE TRANSFERRED)
            IF(NXT1S.LE.NLEFT) GO TO 2000
C--
C--           *TRANSFER THE PROPER NUMBER OF 1S
               LTRANS        = LBITS( IFROM(INDEX) ,NLEFT)
               ITO(INDEX)    = OR(ITO(INDEX),LTRANS)
               IFROM(INDEX) = XOR( IFROM(INDEX) ,LTRANS)
C--
C--        *ELSE (ALL 1-BITS IN THIS WORD ARE TO BE TRANSFERRED)
            GO TO 3000
 2000       CONTINUE
C--
C--           *TRANSFER THE ENTIRE WORD
               ITO(INDEX)    = OR(ITO(INDEX),IFROM(INDEX))
               IFROM(INDEX) = 0
C--
C--        *END IF (ALL 1S TEST)
 3000       CONTINUE
C--
C--        *UPDATE NUMBER OF 1S LEFT TO TRANSFER
            NLEFT = NLEFT - NXT1S
C--
C--        *INCREMENT INDEX FOR NEXT WORD OF BIT-VECTOR
            INDEX = INDEX + 1
C--
C--      *END DO (WORD LOOP)
         GO TO 1000
 4000    CONTINUE
C--
C--      *UPDATE 1S COUNTER FOR THESE BIT-VECTORS
         ITO(1)   = ITO(1)   + NONES
         IFROM(1) = IFROM(1) - NONES
C--
C--      *PERFORM ERROR CHECK TO ENSURE APPROPRIATE NUMBER
C--                                  OF 1-BITS WAS TRANSFERRED
         IF(NLEFT.GT.0) WRITE(6,9001)NONES,IFROM(1),NLEFT
C--
   RETURN
 9001 FORMAT("0$$$$$$$$ TBITSL ERROR - TOO FEW 1-BITS TO TRANSFER",
&   /,       " $$$$$$$$     NONES, IFROM(1), NLEFT = ",3I5)
   END
```

```
C*************************************************************
   SUBROUTINE TBITSR(NONES,IFROM,ITO)
C*************************************************************
C** TBITSR     - TRANSFER 1-BITS FROM RIGHT OF A BIT-VECTOR.
C***       TBITSR IS A FORTRAN SUBROUTINE WHICH WILL TRANSFER
C*** A SPECIFIED NUMBER OF 1-BITS IN THE RIGHTMOST PORTION OF
C*** A BIT-VECTOR, 'IFROM', TO THE CORRESPONDING POSITIONS
C*** OF A BIT-VECTOR, 'ITO'. THE TRANSFERRED BITS ARE THEN
C*** ZEROED OUT IN 'IFROM'. THESE BIT-VECTORS ARE KEPT IN
C*** ARRAYS, ORGANIZED IN THE FOLLOWING MANNER - THE FIRST WORD
C*** OF THE ARRAY CONTAINS THE CURRENT NUMBER OF 1-BITS IN THE
C*** BIT-VECTOR, AND THE REMAINING WORDS OF THE ARRAY CONTAIN THE
C*** ACTUAL BIT-VECTOR.
C***
C*** INPUT -
C***   NONES      - NUMBER OF 1-BITS TO BE TRANSFERRED. NOTE THAT
C***                NONES MUST NOT BE GREATER THAN THE NUMBER OF
C***                1'S IN THE BIT STRING, 'IFROM.
C*** INPUT/OUTPUT -
C***   IFROM      - ARRAY CONTAINING THE BIT-VECTOR FROM WHICH
C***                THE RIGHTMOST 1-BIT POSITIONS ARE TO BE
C***                TRANSFERRED. NOTE THAT THESE RIGHTMOST
C***                1-BITS ARE ZEROED OUT IN 'IFROM' UPON
C***                COMPLETION OF THIS ROUTINE. IFROM(1) IS
C***                A COUNTER WHICH INDICATES THE CURRENT
C***                NUMBER OF 1-BITS IN THE BIT-VECTOR. THE
C***                ACTUAL BIT-VECTOR IS THE CONSECUTIVE BITS
C***                CONTAINED IN THE WORDS IFROM(2)-IFROM(LENGTH)
C***    ITO       - ARRAY CONTAINING THE BIT-VECTOR TO WHICH
C***                THE RIGHTMOST 1-BITS IN 'IFROM' ARE TO BE
C***                TRANSFERRED. THE LEFTMOST POSITIONS IN ITO
C***                REMAIN AS BEFORE. ITO(1) IS A COUNTER WHICH
C***                INDICATES THE CURRENT NUMBER OF 1-BITS IN
C***                THE BIT-VECTOR. THE ACTUAL BIT VECTOR IS THE
C***                CONSECUTIVE BITS CONTAINED IN THE WORDS
C***                        ITO(2) - ITO(LENGTH)
C*** COMMON INPUT -
C***   LENGTH     - LENGTH (IN COMPUTER WORDS) OF THE ARRAYS
C***                CONTAINING THE VARIOUS BIT-VECTORS; IFLYVC, ETC
C*************************************************************
C---
      PARAMETER MAXAC=108,MAXBIT=36,MAXVEC=2+(MAXAC-1)/MAXBIT
      COMMON /ACSTATE/ LENGTH,NACVC(MAXVEC), IFLYVC(MAXVEC),
&                    MAINVC(MAXVEC),NORSVC(MAXVEC), LOSTVC(MAXVEC)
      DIMENSION IFROM(1), ITO(1)
C---
C---      *INITIALIZE TRANSFER LOOP
           NLEFT  = NONES
           INDEX  = LENGTH
C---
C---      *DO WHILE(ALL APPROPRIATE WORDS HAVE NOT BEEN MODIFIED)
 1000      CONTINUE
           IF(NLEFT.LE.0)      GO TO 4000
```

```
          IF(INDEX .LE. 1) GO TO 4000
C---
C---        *COUNT NUMBER OF 1'S IN NEXT WORD
             NXT1S  = N1BITS( IFROM(INDEX) )
C---
C---        *IF(NOT ALL 1-BITS IN THIS WORD SHOULD BE TRANSFERRED)
             IF(NXT1S.LE.NLEFT) GO TO 2000
C---
C---          *TRANSFER THE PROPER NUMBER OF 1'S
               NTRANS        = LBITS( IFROM(INDEX) ,NXT1S-NLEFT)
               ITO(INDEX)    = OR(ITO(INDEX),XOR(IFROM(INDEX),NTRANS))
               IFROM(INDEX) = NTRANS
C---
C---        *ELSE (ALL 1-BITS IN THIS WORD ARE TO BE TRANSFERRED)
             GO TO 3000
 2000        CONTINUE
C---
C---          *TRANSFER THE ENTIRE WORD
               ITO(INDEX)    =  OR (ITO(INDEX),IFROM(INDEX))
               IFROM(INDEX) = 0
C---
C---        *END IF (ALL 1'S TEST)
 3000        CONTINUE
C---
C---        *UPDATE NUMBER OF 1'S LEFT TO TRANSFER
             NLEFT = NLEFT - NXT1S
C---
C---        *DECREMENT INDEX FOR NEXT WORD OF BIT-VECTOR
             INDEX = INDEX - 1
C---
C---      *END DO (WORD LOOP)
          GO TO 1000
 4000     CONTINUE
C---
C---      *UPDATE 1'S COUNTER FOR THESE BIT-VECTORS
          ITO(1)   = ITO(1)   + NONES
          IFROM(1) = IFROM(1) - NONES
C---
C---      *PERFORM ERROR CHECK TO ENSURE APPROPRIATE NUMBER
C---                                  OF 1-BITS WAS TRANSFERRED
          IF(NLEFT.GT.0) WRITE(6,9001)NONES,IFROM(1),NLEFT
C---
  RETURN
 9001 FORMAT("0$$$$$$$$ TBITSR ERROR - TOO FEW 1-BITS TO TRANSFER",
&   /,       " $$$$$$$$     NONES, IFROM(1), NLEFT = ",3I5)
  END
```

```
C*****************************************************************
   SUBROUTINE UEUPDAT(NAC)
C*****************************************************************
C** UEUPDAT    - UPDATE UE-STRENGTH FOR SCENARIO.
C***     THIS ROUTINE IS USED TO UPDATE THE UE-STRENGTH OF THE
C*** SCENARIO. THE UE-STRENGTH IS INITIALIZED AT THE START OF
C*** THE FLYING SCENARIO AND NORMALLY DOES NOT CHANGE THROUGHOUT
C*** THE SIMULATION. HOWEVER, IF THE USER HAS SELECTED THE
C*** AUGMENTATION MODE FOR RESERVE AIRCRAFT, AND ENOUGH RESERVES
C*** ARE AVAILABLE TO MORE THAN REPLACE COMBAT LOSSES, THEN THE
C*** EXCESS RESERVES ARE USED TO INCREASE THE CURRENT UE-STRENGTH.
C***     INCREASING THE UE-STRENGTH REQUIRES RECALCULATION
C*** OF THE LENGTH OF THE AIRCRAFT-STATUS BIT-VECTORS, AND ALSO
C*** REINITIALIZATION OF THE TOTAL-AIRCRAFT-POPULATION VECTOR.
C***
C*** INPUT --
C***   NAC        - NEW UPDATED UE-STRENGTH.
C***   MAXAC      - PARAMETER INDICATING MAXIMUM ALLOWABLE
C***                AIRCRAFT IN THE CURRENT SGM CONFIGURATION.
C***   MAXBIT     - NUMBER OF BITS PER COMPUTER WORD.
C*** COMMON OUTPUT --
C***   LENGTH     - LENGTH (IN COMPUTER WORDS) OF THE AIRCRAFT
C***                BIT-VECTORS. LENGTH MUST BE LARGE ENOUGH SO THAT
C***                THE BIT-VECTORS ARE AT LEAST "NAC" BITS
C***                LONG AT "MAXBIT" BITS PER COMPUTER WORD.
C***   NAC(I)     - (I=1,2,...,LENGTH) BIT-VECTOR WITH FIRST "NAC"
C***                BITS SET TO 1. THIS BIT-VECTOR INDICATES THE
C***                TOTAL AIRCRAFT POPULATION FOR THE SCENARIO.
C*****************************************************************
C---
      PARAMETER MAXBIT=36, MAXAC=108, MAXVEC=2+(MAXAC-1)/MAXBIT
      COMMON /BITS/MASK0,MASK(35), MLEFT0,MSKLFT(36),IZCOUT,
&              ICOUNT(63)
      COMMON /ACSTATE/ LENGTH,NACVC(MAXVEC), IFLYVC(MAXVEC),
&                    MAINVC(MAXVEC),NORSVC(MAXVEC), LOSTVC(MAXVEC)
C---
C--- *PERFORM ERROR CHECK TO ENSURE NO UE-OVERFLOW
      IF(NAC.LE.MAXAC)GO TO 100
        WRITE(6,9001)NAC,MAXAC
        NAC = MAXAC
  100 CONTINUE
C---
C--- *RECOMPUTE LENGTH OF AIRCRAFT-STATUS BIT-VECTORS
      LENGTH = 2 + (NAC-1)/MAXBIT
C---
C--- *INITIALIZE TOTAL-AIRCRAFT-POPULATION BIT-VECTOR
      CALL SPRAY(MSKLFT(36),NACVC(2),LENGTH-1)
      NACVC(LENGTH)=MSKLFT(MOD(NAC-1,MAXBIT))+1
      NACVC(1)=NAC
C---
   RETURN
 9001 FORMAT("0$$$$$$$$ UEUPDAT ERROR - UE OVERFLOW",/,
&          " $$$$$$$$     NAC, MAXAC = ",2I5,/,
&            " $$$$$$$$     NAC WILL BE TRUNCATED TO MAXAC")
    END
```

```
C*************************************************************
   SUBROUTINE WCDIST(NBRKAC,PBRKSEQ,INUXWC,IACVC)
C*************************************************************
C++ WCDIST    - DETERMINE BREAK DISTRIBUTION INTO WORK CENTERS.
C***      WCDIST IS A FORTRAN SUBROUTINE WHICH SIMULATES THE PROCESS
C*** OF AIRCRAFT BREAKING INTO WORKCENTERS. GIVEN A NUMBER OF AIRCRAFT
C*** WHICH HAVE BROKEN INTO AT LEAST ONE WORKCENTER - 1,2,...,NWC, THIS
C*** ROUTINE DETERMINES (BY SIMULATION) WHICH PARTICULAR WORKCENTERS
C*** THE AIRCRAFT BROKE INTO.
C***
C*** INPUTS --
C***   IACVC      - BIT VECTOR INDICATING AIRCRAFT FROM WHICH BROKEN
C***                AIRCRAFT ARE TO BE SELECTED. A 1-BIT
C***                INDICATES AN AIRCRAFT WHICH IS A CANDIDATE FOR
C***                ONE OF THE BROKEN AIRCRAFT. THIS ROUTINE ARBITRARILY
C***                SELECTS THE LEFTMOST 1-BITS AS THOSE AIRCRAFT WHICH
C***                WILL BREAK INTO MAINTENANCE. NORMALLY, IACVC
C***                IS 'IFLYVC' OR 'NORSVC'.
C***   NBRKAC     - NUMBER OF BROKEN AIRCRAFT WHICH ARE TO BE BROKEN
C***                INTO THE DIFFERENT WORKCENTERS. THE LEFTMOST
C***                'NBRKAC' 1-BITS IN 'IACVC' ARE SELECTED AS THE
C***                AIRCRAFT WHICH BROKE.
C*** COMMON INPUTS --
C***   NWC        - TOTAL NUMBER OF WORKCENTERS
C***   MASK(I)    - CONTAINS A 1 IN THE ITH BIT (COUNTING FROM THE LEFT)
C***                AND ZEROES EVERYWHERE ELSE. I=0,...,35
C*** COMMON INPUTS/OUTPUTS --
C***   INREPR(J)  - NUMBER OF AIRCRAFT IN WORKCENTER-J.
C***    LISTRP(I,J) - LISTRP( . ,J) IS A LIST OF AIRCRAFT NUMBERS
C***                 INDICATING THOSE AIRCRAFT REQUIRING MAINTENANCE IN
C***                 THE JTH WORK-CENTER (J=1,2,...,NWC). THIS LIST
C***                 CONTAINS EXACTLY INREPR(J) AIRCRAFT NUMBERS. TO SAVE
C***                 SPACE, THESE LISTS HAVE BEEN PACKED INTO BIT-FIELDS
C***                 INSTEAD OF WORDS. EACH NUMBER IS STORED IN A BIT-FIELD
C***                 "LFLD" BITS WIDE; HENCE, IF "MAXBIT" IS THE LENGTH
C***                 OF A COMPUTER WORD ON THIS SYSTEM, THEN THERE ARE
C***                 (MAXBIT/LFLD) BIT-FIELDS STORED PER WORD. THE AIRCRAFT
C***                 NUMBERS STORED IN THESE BIT-FIELDS INDICATE A UNIQUE
C***                 BIT-POSITION IN THE VARIOUS AIRCRAFT-STATUS BIT-
C***                 VECTORS. THE AIRCRAFT ARE NUMBERED,LEFT-TO-RIGHT,
C***                 0,1,2,...,(MAXAC-1) . TO GET THE ITH AIRCRAFT NUMBER
C***                 IN A WORK-CENTER LIST, THE CORRESPONDING
C***                 BIT-POSITION AND WORD-INDEX MUST BE COMPUTED.
C*************************************************************
C---
      PARAMETER MAXWC=25,MAXBIT=36,MAXAC=108,MAXVEC=2+(MAXAC-1)/MAXBIT
      PARAMETER LFLD=7,NPERWRD=MAXBIT/LFLD,MXINWC=1+(MAXAC-1)/NPERWRD
      COMMON /RSEED/   SEED
      COMMON /WCINPUT/ NWC, NCREWS(MAXWC), SRATE(MAXWC)
      COMMON /WCMAINT/ LISTRP(MXINWC,MAXWC), INREPR(MAXWC)
      COMMON /BITS/ MASKO,MASK(35), MLEFTO,MSKLFT(36),
&                              IZCOUT,ICOUNT(63)
      COMMON /ACSTATE/ LENGTH,NACVC(MAXVEC), IFLYVC(MAXVEC),
&                  MAINVC(MAXVEC),NORSVC(MAXVEC), LOSTVC(MAXVEC)
      DIMENSION PBRKSEQ(2,MAXWC), INDXWC(1), IACVC(1)
C--
C--        *IF(THERE ARE ANY BROKEN AIRCRAFT)THEN
            IF(NBRKAC.EQ.0) GO TO  700
C--
C--          *INITIALIZE NUMBER OF SELECTED AIRCRAFT TO NONE
              NSELEC = 0
C--
C--          *DO FOR(EACH WORD OF THE INPUT AIRCRAFT VECTOR)
              DO 500 IWORD = 2,LENGTH
C--
C--            *INITIALIZE DO
                IACBIT = IACVC(IWORD)
                IBRKVC = 0
C--
C--            *DO FOR(EACH BIT OF THIS BIT-VECTOR WORD)
                DO 400  IBIT = 1,MAXBIT
C--
C--              *IF(THIS BIT INDICATES AN ELIGIBLE AIRCRAFT)THEN
                  MASKAC = MASK(IBIT-1)
                  IF(AND(IACBIT,MASKAC) .EQ. 0) GO TO 300
C--
C--                *SELECT THIS AIRCRAFT AS BROKEN
                    IBRKVC = OR(IBRKVC,MASKAC)
                    NSELEC = NSELEC + 1
C--
C--                *COMPUTE AIRCAFT #
                    IAC = (IWORD-2)*MAXBIT + (IBIT-1)
C--
C--                *DO FOR(EACH WORKCENTER)
                    DO  100  J = 1,NWC
C--
C--                  *DRAW RANDOM SAMPLE FROM UNIFORM (0,1)
                      RDRAW = UNIFM1(SEED)
C--
C--                  *CONTINUE LOOP WITH NEXT WORKCENTER IF
C--                   DRAW INDICATES NO BREAK INTO THIS WC
                      IF(RDRAW .GT. PBRKSEQ(2,J)) GO TO 100
C--
C--                  *UPDATE NUMBER/DISTRIBUTION FOR THIS WC
                      JREAL = INDXWC(J)
                      NTOREP = INREPR(JREAL) + 1
C--
C--                  *MAKE 'IRAND' A RANDOM INTEGER BETWEEN 1 AND
C--                    THE NO. OF A/C THAT WILL BE IN THIS W/C
                      IRAND = INT(UNIFM1(SEED)*FLOAT(NTOREP)) + 1
C--
C--                  *MOVE THE A/C CURRENTLY AT SPOT 'IRAND' IN THE
C--                    LIST TO THE RIGHTMOST SPOT.
                      FLD(MOD(NTOREP-1,NPERWRD)*LFLD,LFLD,
&                          LISTRP(1+(NTOREP-1)/NPERWRD,JREAL))
&                       = FLD(MOD(IRAND-1,NPERWRD)*LFLD,LFLD,
&                              LISTRP(1+(IRAND-1)/NPERWRD,JREAL))
C--
C--                     *INSERT THE SELECTED A/C INTO SPOT 'IRAND'
                         FLD(MOD(IRAND-1,NPERWRD)*LFLD,LFLD,
&                            LISTRP(1+(IRAND-1)/NPERWRD,JREAL))
&                          = IAC
C--
C--                     *INCREMENT THE NO. OF A/C IN THIS W/C
                         INREPR(JREAL) = INREPR(JREAL) + 1
C--
C--                     *EXIT WORKCENTER LOOP IF DRAW ALSO INDICATES
C--                      NO BREAKS INTO REMAINING WORKCENTERS
                         IF(RDRAW .LE. PBRKSEQ(1,J)) GO TO 200
C--
C--                  *END DO (WORKCENTER LOOP)
  100                 CONTINUE
C--
C--              ***** ERROR ***** IF THIS STATEMENT IS REACHED,
C--                                THEN EITHER PBRKSEQ(1,NWC) DOES
C--                                NOT EQUAL 1.0, OR THE
C--                                RANDOM DRAW IS GREATER THAN 1.0
                    WRITE(6,9001)PBRKSEQ(1,NWC),RDRAW
C--
C--                  *EXIT DO (WORKCENTER LOOP) - THIS IS THE
C--                   NORMAL EXIT FROM THE WORKCENTER LOOP
  200                 CONTINUE
C--
C--                  *IF ALL THE BROKEN AIRCRAFT HAVE BEEN SELECTED
C--                   THEN ALL LOOPS ARE TERMINATED
                      IF(NSELEC .GE. NBRKAC) GO TO 600
C--
C--              *END IF (FLYABLE AIRCRAFT TEST)
  300             CONTINUE
C--
C--          *END DO (BIT LOOP)
  400         CONTINUE
C--
C--          *UPDATE INPUT BIT-VECTOR
              IACVC(IWORD)  = XOR(IACVC(IWORD),IBRKVC)
C--
C--      *END DO (WORD LOOP)
  500     CONTINUE
C--
C--              ***** ERROR ***** IF THIS STATEMENT IS REACHED,
C--                                THEN NOT ENOUGH BROKEN AC WERE FOUND
                    WRITE(6,9002)NBRKAC,IACVC(1),NSELEC
C--
C--      *EXIT - ALL BROKEN AC HAVE BEEN SELECTED
  600     CONTINUE
C--
C--      *UPDATE LAST WORD OF INPUT VECTOR
          IACVC(IWORD)  = XOR(IACVC(IWORD),IBRKVC)
C--
```

```
C--          *UPDATE NUMBER OF AIRCRAFT IN INPUT VECTOR
              IACVC(1)  = IACVC(1)  - NBRKAC
C--
C--        *END IF (ZERO BREAKS TEST)
  700       CONTINUE
C--
   RETURN
 9001 FORMAT("0$$$$$$$$$ WCDIST ERROR - SEQUENTIAL SAMPLING ERROR",/,
&            " $$$$$$$$     PBRKSEQ(1,NWC), RDRAW = ",2F10.4)
 9002 FORMAT("0$$$$$$$$$ WCDIST ERROR - INCONSISTENT BROKEN AIRCRAFT",
&            " $$$$$$$$     NBRKAC, IACVC(1), NSELEC = ",3I5)
   END
```

```
C***************************************************************
   SUBROUTINE WCPROB(NWC,PBRKWC,PBRKSEQ,INDXWC,PWCPROD)
C***************************************************************
C** WCPROB     - INITIALIZE WORK-CENTER SEQUENTIAL BREAK PROBABILITIES.
C***       THIS ROUTINE CALCULATES THE PROBABILITIES NECESSARY TO
C***  SIMULATE THE DISTRIBUTION OF AIRCRAFT BREAKS INTO THE VARIOUS
C***  WORKCENTERS.
C***
C*** INPUTS -
C***      NWC       - NUMBER OF WORKCENTERS BEING MODELED.
C***      PBRKWC(J)  - PROBABILITY THAT AN AIRCRAFT WILL BREAK INTO
C***                  WORKCENTER-J. NOTE THAT THIS BREAK MAY BE DUE
C***                  TO SORTIE BREAKS OR GROUND-ABORTS, DEPENDING
C***                  ON HOW THIS ROUTINE IS CALLED.
C***  OUTPUTS -
C***      PBRKSEQ(1,J)- PROBABILITY THAT AN AIRCRAFT BREAKS INTO THE
C***                  WORKCENTER INDICATED BY 'INDXWC(J)', AND DOES
C***                  NOT BREAK INTO ANY OF THE WORKCENTERS -
C***                    INDXWC(J+1),INDXWC(J+2), ..., INDXWC(NWC)
C***                  GIVEN THAT THE AIRCRAFT HAS BROKEN INTO AT LEAST
C***                  ONE OF THE WORKCENTERS -
C***                    INDXWC(J), INDXWC(J+1), ..., INDXWC(NWC).
C***                  THUS, PBRKSEQ(1,(NWC) MUST EQUAL 1.0 .
C***      PBRKSEQ(2,J)- PROBABILITY THAT AN AIRCRAFT HAS BROKEN INTO THE
C***                  WORKCENTER INDICATED BY 'INDXWC(J)', GIVEN THAT
C***                  THE AIRCRAFT HAS BROKEN INTO AT LEAST ONE OF THE
C***                  WORKCENTERS INDICATED BY -
C***                    INDXWC(J), INDXWC(J+1), ..., INDXWC(NWC).
C***      INDXWC(J)  - INDICATES THE INDEX OF THE WORKCENTER WITH THE
C***                  JTH LARGEST BREAK PROBABILITY. THUS, INDXWC(1)
C***                  INDICATES THE WORKCENTER WITH THE LARGEST
C***                  BREAK PROBABILITY, INDXWC(NWC) INDICATES THE
C***                  ONE WITH THE SMALLEST, ETC. .
C***      PWCPROD    - PRODUCT-FORMULA OVERALL WORK-CENTER BREAK-RATE.
C***                  COMPUTED FROM THE INDIVIDUAL WC BREAK-RATES.
C***************************************************************
C--
     DIMENSION PBRKWC(NWC), PBRKSEQ(2,NWC), INDXWC(NWC)
C--
C--  *COMPUTE SORTED ARRAY OF WORK-CENTER INDICES ACCORDING TO
C--   LARGEST-TO-SMALLEST BREAK-RATE.
      CALL SORTDS(NWC+0,PBRKWC,INDXWC)
C--
C--  *INITIALIZE END-POINT PROBABILITIES
      PBRKSEQ(1,NWC) = 1.0
      PBRKSEQ(2,NWC) = 1.0
C--
C--     *DO UNTIL(ALL PROBABILITIES HAVE BEEN CALCULATED)
         J = NWC - 1
         POLD = 1.0 - PBRKWC(INDXWC(NWC))
 100     CONTINUE
C--
C--        *COMPUTE NEXT PROBABILITIES
```

```
                PROB  = PBRKWC(INDXWC(J))
                PNEW  = POLD * (1.0 - PROB)
                PBRKSEQ(2,J) = PROB/(1.0 - PNEW)
                PBRKSEQ(1,J) = PBRKSEQ(2,J) * POLD
                POLD   = PNEW
C---
C---     *END DO (WC LOOP)
          J = J - 1
          IF(J.GT.0) GO TO 100
C---
C---  *SAVE PRODUCT-FORMULA OVERALL WORK-CENTER BREAK-RATE
       PWCPROD = 1.0 - POLD
C---
   RETURN
   END
```

```
C*************************************************************
      SUBROUTINE WCREAD(IFILE,MAXWC,NWC,PBRKWC,NCREWS,SRATE)
C*************************************************************
C** WCREAD     - READ AND INITIALIZE WORK CENTER DATA.
C***      WCREAD READS WORK-CENTER DATA FROM THE MAINTENANCE
C*** MANPOWER INPUT FILE. THIS DATA IS ASSUMED TO BE ON UNIT
C*** "IFILE", ONE FREE-FORMAT RECORD PER WORK-CENTER.
C*** THIS ROUTINE RETURNS THE NUMBER OF WORK-CENTERS LOADED,"NWC";
C*** THE BREAK-RATE ARRAY, "PBRKWC"; THE SERVICE-RATE ARRAY, "SRATE";
C*** AND THE SERVERS ARRAY, "NCREWS".
C***      THE SERVERS ARRAY, "NCREWS", REPRESENTS THE NUMBER OF CREWS
C*** AVAILABLE PER 12-HOUR SHIFT.  THE INPUT FILE CONTAINS THE
C*** TOTAL NUMBER OF CREWS AVAILABLE FOR THE PARTICULAR WORK-CENTER.
C*** THE 12-HOUR SHIFT NUMBER IS COMPUTED BY DIVIDING THIS AVAILABLE-
C*** SERVERS IN HALF AND ROUNDING TO THE NEAREST INTEGER NUMBER.
C*** IN ADDITION, IT IS ASSUMED THERE IS ALWAYS AT LEAST ONE CREW
C*** PER SHIFT.
C***      THE AFSC DESCRIBING THE WORK-CENTER, AND THE TOTAL NUMBER OF
C*** SERVERS ARE ECHO-PRINTED, BUT ARE NOT SAVED FOR FUTURE USE.
C***      THE MAXIMUM NUMBER OF WORK-CENTERS WHICH CAN BE LOADED IS
C*** SPECIFIED BY THE "MAXWC" INPUT PARAMETER. IF MORE THAN
C*** THIS NUMBER IS READ, AN ERROR MESSAGE IS PRINTED, AND THE SGM
C*** RUN CONTINUES WITH ONLY THE FIRST "MAXWC" WORK-CENTERS.
C*** INPUTS --
C***       IFILE    - INPUT FILE NUMBER FROM WHICH MAINTENANCE MANPOWER
C***                  INPUT DATA IS TO BE READ
C***       MAXWC    - MAXIMUM NUMBER OF WORK-CENTERS WHICH CAN BE LOADED
C*** OUTPUTS --
C***       NWC      - NUMBER OF WORK-CENTERS LOADED
C***       PBRKWC   - ARRAY OF WORK-CENTER BREAK RATES
C***       NCREWS   - ARRAY OF WORK-CENTER CREW NUMBERS
C***       SRATE    - ARRAY OF WORK-CENTER SERVICE RATES
C*************************************************************
C---
      DIMENSION PBRKWC(MAXWC), NCREWS(MAXWC), SRATE(MAXWC)
      CHARACTER CAFSC*5
C---
C---  *PRINT HEADER FOR ECHO-CHECK OF INPUT DATA
       WRITE(6,9001)
C---
C---  *INITIALIZE NUMBER OF WORK CENTERS
       NWC=1
C---
C---  *DO UNTIL(NO MORE WORK CENTERS TO LOAD)
  100  CONTINUE
C---
C---     *READ NEXT WORK-CENTER RECORD
          READ(IFILE,9000,END=200)
&                  CAFSC,PBRKWC(NWC),SERVERS,SRATE(NWC)
C---
C---     *PERFORM ERROR CHECK ON INPUT DATA
          IF((PBRKWC(NWC).GT.0.0).AND.(PBRKWC(NWC).LE.1.0))
&                                   GO TO 150
```

```
        IF(SERVERS.GE.0.0)GO TO 150
        IF(SRATE(NWC).GE.0.0)GO TO 150
          WRITE(6,9004)CAFSC,PBRKWC(NWC),SERVERS,SRATE(NWC)
          GO TO 100
 150    CONTINUE
C---
C---     *COMPUTE INTEGER-NUMBER OF CREWS PER 12-HOUR SHIFT
        NCREWS(NWC)=MAX0(1,INT(SERVERS*.5 + .5))
C---
C---     *ECHO-PRINT WORK-CENTER INFORMATION
        WRITE(6,9002)NWC,CAFSC,PBRKWC(NWC),SERVERS,SRATE(NWC)
C---
C---     *INCREMENT WORK-CENTER INDEX
        NWC=NWC+1
C---
C---  *END DO (WORK-CENTER LOOP)
      IF(NWC.LE.MAXWC)GO TO 100
C---     *PRINT ERROR MESSAGE IF STILL MORE DATA ON THE FILE
        READ(IFILE,9000,END=200)PBRKWC(NWC),SERVERS,SRATE(NWC)
        WRITE(6,9003)MAXWC
 200  CONTINUE
C---
C---  *ADJUST NUMBER OF WORK-CENTERS TO ACCOUNT FOR EOF READ
      NWC=NWC-1
C---
C---  *CLOSE-OUT MANPOWER INPUT FILE
      CALL FCLOSE(IFILE)
C---
  RETURN
 9000 FORMAT(1X,A5,1X,F9.4,1X,F9.2,1X,F9.4)
 9001 FORMAT("1",//,7X,"*********************************************",/,
&                 ,7X,"**********  AIRCRAFT MAINTENANCE  **********",/,
&                 ,7X,"*********************************************",//,
&   21X," BREAK",1X,"  TOTAL",3X,"SERVICE RATE",/,
&   7X,"WC #",2X," AFSC",3X," RATE",2X," SERVERS",2X,"(ACFT/HOUR)",//)
 9002 FORMAT(7X,I3,3X,A5,3X,F6.4,1X,F7.2,3X,F9.4)
 9003 FORMAT("0$$$$$$$$ WCREAD ERROR - TOO MANY WORK-CENTERS ",
&   "IN THE INPUT FILE",/,
&   " $$$$$$$$      ONLY THE FIRST ",I3," WORK-CENTERS WERE USED:",/,
&   " $$$$$$$$      INCREASE -MAXWC- PARAMETER IF YOU WANT MORE WC-S")
 9004 FORMAT("0$$$$$$$$ WCREAD ERROR - INVALID WORK CENTER DATA",/,
&            " $$$$$$$$      PBRKWC, SERVERS, SRATE = ",3F8.3)
  END
```

```
C*****************************************************************
   REAL FUNCTION XNORM (XMEAN,STDEV,SEED)
C*****************************************************************
C++ XNORM      - DRAW RANDOM SAMPLE FROM A NORMAL DISTRIBUTION.
C***   THIS IS A REAL-VALUED FORTRAN FUNCTION WHICH GENERATES
C*** A RANDOM SAMPLE ACCORDING TO A NORMAL PROBABILITY
C*** WITH THE GIVEN INPUT MEAN AND STANDARD DEVIATION.
C*** THE TECHNIQUE IS TO APPROXIMATE A NORMAL DISTRIBUTION USING
C*** THE CENTRAL LIMIT THEOREM. 12 INDEPENDENT SAMPLES ARE
C*** DRAWN FROM A UNIFORM(0,1) DISTRIBUTION AND THEN ADDED.
C*** THE RESULT IS APPROXIMATELY NORMALLY DISTRIBUTED WITH MEAN 6
C*** AND STANDARD DEVIATION 1. THE SAMPLE IS THEN TRANSLATED TO
C*** OBTAIN A SAMPLE FROM A DISTRIBUTION WITH THE GIVEN
C*** INPUT MEAN AND STANDARD DEVIATION.
C***
C*** INPUTS --
C***   XMEAN    - MEAN OF THE NORMAL DISTRIBUTION FROM WHICH
C***              THE SAMPLE IS TO BE GENERATED.
C***   STDEV    - STANDARD DEVIATION OF NORMAL DISTRIBUTION FROM
C***              WHICH SAMPLE IS TO BE GENERATED. IF THIS VALUE
C***              IS NEGATIVE, AN ERROR MESSAGE IS PRINTED.
C*** INPUT/OUTPUT --
C***   SEED     - CURRENT SEED OF RANDOM NUMBER GENERATOR.
C*** OUTPUT --
C***   XNORM    - RANDOM SAMPLE FROM A NORMAL DISTRIBUTION WITH
C***              GIVEN MEAN AND STANDARD DEVIATION.
C*****************************************************************
C---
C---  *IF(STANDARD DEVIATION IS LEGITIMATE)THEN
      IF(STDEV.LT.0.0)GO TO 100
C---
C---     *DRAW SAMPLES FROM 12 UNIFORM (0,1) DISTRIBUTIONS AND
C---            ADJUST THE MEAN OF THIS RANDOM SAMPLE TO ZERO
         XNORM=UNIFM1(SEED)+UNIFM1(SEED)+UNIFM1(SEED)+UNIFM1(SEED)
&            +UNIFM1(SEED)+UNIFM1(SEED)+UNIFM1(SEED)+UNIFM1(SEED)
&            +UNIFM1(SEED)+UNIFM1(SEED)+UNIFM1(SEED)+UNIFM1(SEED)-6.
C---
C---     *CONVERT TO A SAMPLE FROM DISTRIBUTION WITH APPROPRIATE
C---                   MEAN AND STANDARD DEVIATION.
         XNORM = XMEAN + STDEV*XNORM
C---
C---  *ELSE (NEGATIVE STANDARD DEVIATION)
      GO TO 200
  100 CONTINUE
C---
C---     *SET RETURN VALUE TO ZERO AND PRINT ERROR MESSAGE
         XNORM = 0.0
         WRITE(6,9001)STDEV
C---
C---  *END IF (STD DEV TEST)
  200 CONTINUE
C---
   RETURN
```

```
 9001 FORMAT("0$$$$$$$$$ XNORM ERROR - NEGATIVE STANDARD DEVIATION",/,
&           " $$$$$$$$     STDEV = ",F10.5)
   END
```

```
C*************************************************************
   SUBROUTINE ZBITSL(NONES,IARRAY)
C*************************************************************
C** ZBITSL     - ZERO-OUT 1-BITS IN LEFTMOST PORTION OF A WORD.
C***       ZBITSL IS A FORTRAN SUBROUTINE WHICH WILL ZERO-OUT
C*** A SPECIFIED NUMBER OF 1-BITS IN THE LEFTMOST PORTION OF
C*** A BIT-VECTOR. THIS BIT-VECTOR IS KEPT IN AN ARRAY,
C*** ORGANIZED IN THE FOLLOWING FASHION - THE FIRST WORD OF
C*** THE ARRAY CONTAINS THE CURRENT NUMBER OF 1-BITS IN THE
C*** BIT-VECTOR, AND THE REMAINING WORDS OF THE ARRAY CONTAIN
C*** THE ACTUAL BIT-VECTOR. THIS ROUTINE ZEROES OUT THE PROPER
C*** 1-BITS, AND UPDATES THE 1-BIT COUNTER IN THE FIRST WORD
C*** OF THE ARRAY.
C***
C*** INPUT -
C***   NONES      - NUMBER OF 1-BITS TO BE ZEROED-OUT. NOTE THAT
C***                NONES MUST BE .LE. NUMBER OF 1S IN THE BIT-
C***                VECTOR.
C*** INPUT/OUTPUT -
C***   IARRAY     - ARRAY CONTAINING THE BIT-VECTOR TO BE
C***                MODIFIED. IARRAY(1) IS A COUNTER WHICH INDICATES
C***                THE CURRENT NUMBER OF 1-BITS IN THE BIT-VECTOR.
C***                THE ACTUAL BIT-VECTOR IS THE CONSECUTIVE BITS
C***                CONTAINED IN THE WORDS IARRAY(2)-IARRAY(LENGTH)
C*** COMMON INPUT -
C***   LENGTH     - LENGTH (IN COMPUTER WORDS) OF THE ARRAYS
C***                CONTAINING THE VARIOUS BIT-VECTORS; IFLYVC, ETC
C*************************************************************
C---
      PARAMETER MAXAC=108, MAXBIT=36, MAXVEC=2+(MAXAC-1)/MAXBIT
      COMMON /ACSTATE/ LENGTH,NACVC(MAXVEC), IFLYVC(MAXVEC),
&                    MAINVC(MAXVEC),NORSVC(MAXVEC), LOSTVC(MAXVEC)
      DIMENSION IARRAY(1)
C---
C---       *INITIALIZE DO
            NLEFT  = NONES
            INDEX  = 2
C---
C---       *DO WHILE(ALL APPROPRIATE WORDS HAVE NOT BEEN MODIFIED)
 1000       CONTINUE
            IF(NLEFT.LE.0)       GO TO 4000
            IF(INDEX.GT.LENGTH) GO TO 4000
C---
C---          *COUNT NUMBER OF 1S IN NEXT WORD
               NXT1S  = N1BITS( IARRAY(INDEX) )
C---
C---          *IF(NOT ALL 1-BITS IN THIS WORD SHOULD BE ZEROED)
               IF(NXT1S.LE.NLEFT) GO TO 2000
C---
C---             *ZERO-OUT THE APPROPRIATE NUMBER OF 1S
                  IARRAY(INDEX) = XOR( IARRAY(INDEX) ,
&                                  LBITS(IARRAY(INDEX),NLEFT) )
C---
```

```
C--          *ELSE (ALL 1-BITS IN THIS WORD ARE TO BE ZEROED)
              GO TO 3000
 2000         CONTINUE
C--
C--             *ZERO-OUT THE ENTIRE WORD
                 IARRAY(INDEX) = 0
C--
C--          *END IF (ALL 1S TEST)
 3000         CONTINUE
C--
C--          *UPDATE NUMBER OF 1S LEFT TO ZERO-OUT
              NLEFT = NLEFT - NXT1S
C--
C--          *INCREMENT INDEX FOR NEXT WORD OF BIT-VECTOR
              INDEX = INDEX + 1
C--
C--       *END DO (WORD LOOP)
           GO TO 1000
 4000      CONTINUE
C--
C--       *UPDATE 1S COUNTER FOR THIS BIT VECTOR
           IARRAY(1) = IARRAY(1) - NONES
C--
C--       *PERFORM ERROR CHECK TO ENSURE APPROPRIATE NUMBER
C--        OF 1-BITS WAS ZEROED-OUT
           IF(NLEFT.GT.0) WRITE(6,9001)NONES,IARRAY(1),NLEFT
C--
   RETURN
 9001 FORMAT("0$$$$$$$$ ZBITSL ERROR - NOT ENOUGH 1S TO ZERO",/,
&   " $$$$$$$$      NONES,IARRAY(1),NLEFT = ",3I5)
   END
```

*

APPENDIX C

FEDERAL INFORMATION PROCESSING STANDARD SOFTWARE SUMMARY

01. Summary date Yr. Mo. Day 8 1 09 1 0	02. Summary prepared by (Name and Phone) Michael J. Konvalinka (301) 229-1000	03. Summary action New [✓ X] Replacement — Deletion — Previous Internal Software ID
04. Software date Yr. Mo. Day 8 1 09 1 0	05. Software title The Sortie-Generation Model System Volume II Sortie-Generation Model User's Guide	07. Internal Software ID None
06. Short title		

08. Software type	09. Processing mode	10. Application area — General		Specific
[✓ X] Automated Data System	— Interactive	— Computer Systems Support Utility	— Management Business	Logistics Capability Assessment
— Computer Program	— Batch	— Scientific Engineering	— Process Control	
— Subroutine Module	[✓ X] Combination	— Bibliographic Textual	[✓ X] Other	

11. Submitting organization and address	12. Technical contact(s) and phone
Logistics Management Institute 4701 Sangamore Road, P. O. Box 9489 Washington, D.C. 20016	Mr. John B. Abell Mr. Michael J. Konvalinka (301) 229-1000 AV 287-2779

13. Narrative

The Sortie-Generation Model System provides the capability for relating aircraft spares and maintenance manpower levels to the maximal sortie-generation capability of tactical air forces over time.

14. Keywords

Readiness; Resource Allocation; Sortie Generation Capability; Logistics Capability Assessment

15. Computer manuf'r and model	16. Computer operating system	17. Programming language(s)	18. Number of source program statements
Honeywell G-635	GCOS	Cobol 600 Fortran 600/GMAP	15000
19. Computer memory requirements	**20. Tape drives**	**21. Disk/Drum units**	**22. Terminals**
49k words 36 bits each	4	1 Disk	1 time sharing

23. Other operational requirements

24. Software availability			25. Documentation availability		
Available	Limited	In-house only	Available	Inadequate	In-house only
X	—	—	X	—	—

26. FOR SUBMITTING ORGANIZATION USE

STANDARD FORM 185
JULY 1974
U.S. DEPT. COMMERCE NBS
FIBS PUB 30

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) The Sortie-Generation Model System Volume II Sortie-Generation Model User's Guide		5. TYPE OF REPORT & PERIOD COVERED
		6. PERFORMING ORG. REPORT NUMBER LMI Task DP101
7. AUTHOR(s) John B. Abell, Robert S. Greenberg Michael J. Konvalinka		8. CONTRACT OR GRANT NUMBER(s) MDA903-80-C-0054
9. PERFORMING ORGANIZATION NAME AND ADDRESS Logistics Management Institute 4701 Sangamore Road, P. O. Box 9489 Washington, D.C. 20016		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Assistant Secretary of Defense (Manpower, Reserve Affairs, & Logistics) The Pentagon, Washington, D.C.		12. REPORT DATE September 1981
		13. NUMBER OF PAGES 135
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report)
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

"A" Approved for public release

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Unlimited

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Readiness; Resource Allocation; Sortie Generation Capability; Logistics Capability Assessment

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Sortie-Generation Model System provides the capability for relating aircraft spares and maintenance manpower levels to the maximal sortie-generation capability of tactical air forces over time.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

DA
FILM